An Adaptable Formalism
for the
Computational Analysis
of
English Noun Phrase Reference

Geraint Anthony Wiggins

PhD
University of Edinburgh
1990

Declaration

I declare that this thesis has been composed by myself and that the research reported therein has been conducted by myself unless otherwise stated.

Geraint Anthony Wiggins

31st August 1990

Acknowledgements

This thesis is dedicated to my parents,
Brian and Dilys Wiggins,
without whose encouragement and support
over the past twenty-eight years
the work would simply not have been done.

Thanks are also due to many other people, but particularly: to Mark Steedman, Ewan Klein and Robin Cooper for their advice as my successive second supervisors; to Barry Richards and Mark Steedman, for persuading me to keep trying; to Judy Delin, Roberto Desimone and Mike Malloch for comradeship in adversity; to Chris Mellish for a number of lengthy and useful discussions, and for giving me the basic idea; to the Science and Engineering Research Council, during whose studentship number 84311901 the majority of this work was carried out; to the Department of AI at Edinburgh, and to Alan Bundy and the DReaM group, for supporting me financially and intellectually for the past three years.

Finally, I am deeply grateful to my long-suffering supervisor, Graeme Ritchie, without whose painstaking, patient and consistent attention to detail this document would be considerably less than it is.

Abstract

This thesis addresses issues of analysis and representation of the meaning of human language on computers.

The notion of *adaptable* representations, which are designed to encode ambiguity implicitly, is introduced, and placed in a context of related work on representation and reference analysis.

A parsing formalism is presented which allows the analysis and translation of a subset of English into a representation language which encodes not only the truth-functional semantics of the input discourse but also some of its surface form. The parsing algorithm works strictly word-by-word, always maintaining a self-contained partial representation of the entire discourse so far, in order to provide a stringent framework for examination of the idea of adaptability. It is shown that the use of adaptable representations which implicitly encode local lexical ambiguity can improve the efficiency of parsing by reducing non-determinism.

A basic noun phrase reference analysis algorithm is presented, which allows inferences to be made about the reference of the input discourse. Following the common belief that noun phrase reference analysis in humans takes place as information is received, processing is performed word-by-word, in parallel with parsing. Part of the reference analysis algorithm constitutes a new view of the distinction between restrictive and non-restrictive noun phrase modifiers. It is shown that this new view and the adaptability of the system in general does not compromise the ability of the reference analysis algorithm to produce correct results.

Finally, a new approach to the generation of readings of sentences containing many-ways ambiguous or vague set reference is presented. The approach relies heavily on adaptability, and it is suggested that without (an equivalent of) adaptability any approach is likely to be combinatorially explosive and therefore in general computationally intractable. This is presented as one specific justification for the more general utility of adaptability of representation.

Contents

3. The George Parsing and DeReferencing System 33

4. The George Representation Language 47

5. The George Parser 83

6. The George DeReferencer 121

Preface

1. This Document

This document is a statement and discussion of the theory and implementation of the George Parsing and DeReferencing system. As such, as well as a problem statement and literature survey, it covers three main headings: the George Parser; the George Representation Language; and the George Noun Phrase DeReferencer.

These three parts of the George system form a unified and closely linked whole; they are not intended to work independently of each other, neither practically nor theoretically. This should be borne in mind when reading the explanations in this document. The closely linked structure has necessitated a certain amount of forward referencing, which has been planned so as to minimise distraction to the reader.

2. Abbreviations

The only common abbreviations used in this document are "FOPC" for "First Order Predicate Calculus" and "GRL" for "George Representation Language". Otherwise, terms are in general spelled out in full.

3. Notation and Special Symbols

There are three kinds of text in this document: explanatory text, program output and theoretical statements. Explanatory text will be printed like this, in plain Classic font. Program output will be boxed, labelled and `printed in titan font, like this`. Important theoretical points will be **emboldened**, in particular in the specification of the George System's reference analysis rules. Specially defined terms appear in *italic*, where it is necessary to emphasise that they have a particular usage; ordinary emphasis will be underlined.

The following non-alphanumeric symbols are used in this document, but do not appear in the George Representation Language.

$+$, $\_^{\text{operation}}\!\!\rightarrow$	combines with, under operation, to give (eg $3+2\ \_^{\text{multiplication}}\!\!\rightarrow\ 6$)
2^S	power set of S
$\{x:P(x)\}$	set of all x such that P(x) is true
$\cup,\cap,\subseteq,\supseteq,\subset,\supset,\in$	the conventional set relations
$\exists$	the conventional existential quantifier
$\leq,=,\geq,>$	the conventional arithmetic relations
(), {}, []	the conventional parentheses/braces/brackets
$\|S\|$	modulus: the number of elements in set S
*	at the left margin marks an ill-formed sentence in English or logic
$\oplus$	logical exclusive or
$\neg, \wedge, \vee, \Rightarrow$	the conventional logical not, and, inclusive or, implies

The following non-alphanumeric symbols are used in this document, and have a special meaning in the George Representation Language, given in the righthand column.

$\forall$	universal quantification	index quantification
$<$	arithmetic less than	index bounding
$\times$	cross product	weak index application

The following non-alphanumeric symbols are used in this document to denote concepts specific to the George Representation Language.

~ (tilde)	sort application
$\otimes$	strong index application
!	definiteness
▸	context extension

Where quotes or variants on quotes from other researchers' work appear in this document, the symbols are to be read as appropriate to the original work; where necessary, definitions will be given.

Chapter 1

Introduction: The Problem of Reference to Underspecified Sets

Abstract

The central problem at which this thesis is aimed is outlined. It is the problem of reference by noun phrases, quantified or otherwise, to sets of individuals in the discourse which have not been fully specified. The particular underspecification discussed here is in terms of number.

1. Introduction

Consider this simple puzzle.

> "Some men own some donkeys. Five donkeys escape. The three donkeys who do not escape stay in the paddock. Each man owns one donkey. The men who own the donkeys who escape are angry.
>
> How many men, how many angry men and how many donkeys are there?" 1

The solutions are not hard to calculate. The numerical inference required to arrive at the answers is well within the scope of existing reasoning systems designed by researchers in the field of Artificial Intelligence.

The more interesting aspect of the puzzle is the understanding of the reference in it. In particular, there are references to sets of objects (*viz* men and donkeys), but the whole point of the puzzle hinges on the fact that these sets are not fully specified – the number of elements in each is left deliberately undefined.

The pattern of events is this. First, two uniform sets, say M and D (for "men" and "donkeys") are introduced. Because the sets are uniform, their elements are not mutually discriminable. There is a relation (ownership) between the members of the two sets, the details of which are not given. Next, the set D is divided into two subsets, say D_{escape} and

D_{rest}. A cardinality is specified for D_{escape}, the members of which can now be distinguished from those of D_{rest}, whose cardinality is still unknown. In the third sentence, such a cardinality is supplied. Finally, the relationship between M and D is specified more fully, to the extent that reasoning about the cardinality of D, based on D_{escape}, D_{rest} and the connection between them and D, allows us to reason also about the cardinality of M.

This document is about a computational solution to the problems raised in the attempt to emulate the behaviour of the human language understanding mechanism as it processes reference of this *underspecified* kind, focussing on the human ability to represent and reason about such reference as it becomes more specified.

2. The Problem of Reference to Underspecified Sets

Many researchers have considered the general problem of noun phrase reference (see [Hirst, 1981] for a comprehensive survey up to 1980). Even so, it is generally agreed that there are still many problems to be dealt with in that area. Particular successes in the field have been Terry Winograd's SHRDLU [Winograd 1971], Bundy *et al*'s MECHO (specifically Chris Mellish's contribution, in this area of interest) [Mellish, 1981], and Bonnie Webber's "Formal Approach to Discourse Anaphora" [Webber, 1979]. Indeed, the bulk of the George system described here rests heavily on the work of these last two authors, whose work is summarised in Chapter 2.

Part of the reason why noun phrase reference is such a large problem is the large variety of forms it can and frequently does take. [Webber, 1979] discusses in detail three particular forms of NP anaphor: Pronominal reference, Definite Noun Phrase reference, and (so-called) One-Anaphora. But these are just three kinds out of a range of ten which Webber lists. In this document, I shall specialise further – looking only at Prominal and Noun Phrase reference, though including indefinite reference in the latter, and suggesting that the two may be dealt with in the same way.

Above and beyond the multiplicity of kinds of noun phrase reference, problems also arise from the tendency in common natural language usage for speakers and writers to leave information out of the initial specification of and subsequent references to an entity or a set of entities taking part in a discourse. These omissions are often filled in later – though, sometimes, they are not filled in at all. They may take place for a number of reasons, some of which are deliberately stylistic – for example, to improve clarity, or to maintain tension. Sometimes they are made because of plain laziness, but in most cases the speaker or writer will use them to improve the efficiency of the communication, leaving some

information to be inferred by the hearer or reader. This inference, it would appear, often occurs as (or at least very soon after) a noun phrase is heard or read.

An example of the use of pronouns to increase bandwidth, where ambiguity is introduced by that use, is the well-known "donkey sentence"[1]:

"Every man who owns a donkey beats it." 2

Translation of the donkey sentence in isolation is indeterminate in two ways. First, and less interestingly for our current purpose, there is ambiguity between the generic reading, where there is no particular set of men, and the referential reading, where such a set has been introduced into the discourse by a previous noun phrase.

The other, more interesting, indeterminate aspect of the translation arises only in the second of these interpretations. It is a kind of vagueness which only matters in certain contexts, and it seems not to have been singled out for specific attention in the past. Consider, for example, the discourse:

"There are ten men. 3a
Five of the men own a donkey each. 3b
Every man who owns a donkey beats it." 3c

We could represent this with the diagram in Figure 1.

Now consider example 4:

"There are ten men. 4a
Some of the men own a donkey each. 4b
Every man who owns a donkey beats it." 4c

Two noteworthy points arise in this second version. First, it is clearly impossible to draw a diagram like figure 1, because the reference of the subject in 4b is *underspecified* – by which I mean that some of the information which it could carry is absent. Specifically, the subject noun phrase of 4b is underspecified in number.

1: I will refer to the points of interest in this sentence – the indefinite singular reference and its associated singular pronoun, "it" – as the "donkey existential" and "donkey pronoun" respectively, to distinguish them from the more general example sentences presented here, almost all of which have something to do with donkeys.

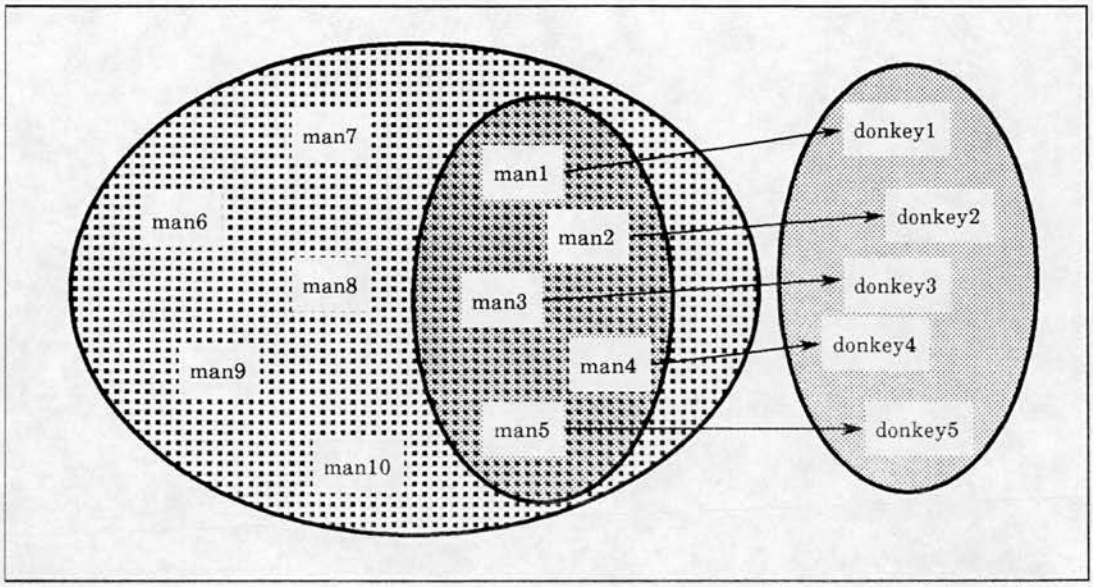


Figure 1: *Ten men and five donkeys*

Secondly – perhaps surprisingly – this makes no difference at all to the human ability to reason about the relationships between the sets in the discourse, until a specific reference is made, for example, to the number of animals, as in 5.

"How many donkeys are there?" 5

We can even make statements as complex as 6, where we refer to subsets of and set differences between subsets of sets mentioned but not enumerated in 4.

"Half of the donkeys are brown; the other half are grey.
The men who own grey donkeys dislike some of the men who own brown ones." 6

While these examples are perhaps a little strained, they serve to show just how far we can go, meaningfully, in a discourse without even trying to resolve this ambiguity, neither from the speaker's nor from the hearer's point of view. Indeed, one approach to dealing with the simple examples of this problem given here so far might be to compare it with so-called lazy evaluation of mathematical expressions in some computer languages. To put this basic idea into a context rather closer to our subject here: maybe a useful approach is to mirror the original ambiguity of the English discourse, in whatever representation of its meaning we use on a computer, from the point at which it occurs in the input as far as possible into the discourse – or until the input is itself no longer ambiguous.

Returning for the moment to a more general level, there is a further problem. The sentences in examples 3 and 4 are special in that the relationship between the elements of the sets is deliberately contrived to be one-to-one in, at least, what I think is the most natural reading. In general, this will not be the case, as can be seen, for example, in 7a-c.

Now, it may well be necessary at some stage in a discourse to state explicit relationships between elements and/or subsets of these underspecified sets of referents, as, for example, in 7d-f.

"There are ten men.	7a
Some of the men own donkeys.	7b
Each man who owns a donkey beats it.	7c
Jim is one of the men.	7d
Jim beats his donkey for fun.	7e
The other men pity Jim's donkey."	7f

If we wish to be able to discuss Jim's donkey as an individual entity, we must have a means to enumerate parts of the sets specified in the discourse, while leaving other parts simply as underspecified (sub)sets. To do this, we need a formal means of separating out the exact relationships between Jim and his donkey as (under)specified in the English discourse. This concept is the central focus of the work presented here.

3. The Timing of Reference Evaluation

In general, when we consider the analysis of reference in anything other than trivial natural language, we are presented with a particular problem of inference: it is almost never clear exactly how far one should go in making inferences about the items which one is discussing and/or the situations in which they are placed. In Figure 1, I illustrated a very simple situation where we can easily take an obvious course of action, and work out everything as far as possible. However, it is certainly not always possible to do this – for example, in cases where there are too many elements in a set to consider them all, or, worse, where we do not know how many elements there are.

It is not immediately clear how one might go about determining what is the "correct" amount of inference to apply to a referring expression; indeed, one might make a strong argument for there being no such quantity, by reference to the easily observable variation in application of this kind of inference between different individuals. There is, however, a minimal, "safest" position, analogous with the ideas of "lazy evaluation" in computer

programming languages: an expression is left unevaluated until its result is explicitly needed. Thus, when we finally analyse a reference we always have as much information as possible; if it is still ambiguous, then there is nothing further we can do, so we are justified in allowing the ambiguity to proceed, in the hope that it will be resolved later in the discourse. While I do not claim that is necessarily a psychologically plausible solution (indeed, it is not so in general), it is useful from a computational point of view, and serves our purposes here. In particular, it will prove useful to apply this idea to the representation of sets of entities in the translations of natural language utterances.

The application of a notion of lazy evaluation begs a question of how we are to proceed to analyse discourse, which may be very heavily ambiguous, and at the same time postpone evaluation of the structures which constitute our data. Also, there are clear advantages, both psycholinguistically and practically, in performing at least some reference evaluation as early as possible – see, for example, the work of [Mellish, 1981], summarised in Chapter 2.

Lazy and early evaluation fit most comfortably into language analysis if we allow ourselves one fundamental underlying assumption. Fortunately, this assumption is not unreasonable. It is simply that the person (or program) generating the input to our analyser intends to make sense. It gives us two advantages.

First, we do not have to make judgements about what is (in some sense) "correct" and what is not. We simply attempt to understand or analyse; we may either succeed or fail. This is in line with the human sentence processing mechanism which does not, on the whole, make correctness judgements about its input, but attempts to derive some meaning from it, even if the input is at odds with expectation – though, of course, we must acknowledge that expectation can play an important (if sometimes misleading) part.

Second, and more importantly, our assumption means that we are free to suppose that reference will at some stage in the course of the discourse become unambiguous, (if it is necessary that it does so. Plenty of ambiguities can often be harmlessly left unresolved – *eg* the collective *vs* distributive ambiguity in ordinary plural noun phrases).

It will be seen later that this assumption affects design decisions in the theory. These decisions are, however, not irreversible: if the "judgemental" view were required, it would not be hard to adapt the theory or the implementation.

A corollary of the assumption is that we want to represent ambiguity (*ie* make all the possible readings available) in our system; otherwise, we will not be able to resolve it

appropriately when the time comes. Many ways of doing so already exist, for example: simple backtracking or breadth-first shift/reduce parsers; chart parsers; word expert parsers. Another example on a different level is Mellish's reference analyser in the MECHO program. I will argue that none of these approaches is adequate for the task in hand here, though [Mellish, 1981]'s work on "early evaluation and incremental analysis" constitutes an unprecedentedly large step in the right direction.

In particular, when set references are involved, really vast numbers of readings may be available from just one predication, because the relations between sets referred to by ordinary plural noun phrases are not specified, except in as far as they exist. Take for example sentence 8.

"Ten men own ten donkeys." 8

It is easy to list large numbers of ways in which the exact details of such an ownership relation could be worked out. (It would be boring to do so here. The point will be raised again in Chapter 7.) Given this, and the ability to refer to arbitrary subsets, individuals, set differences, and so on, and the combinatorially explosive effect of subsequent underspecified reference to the same sets, any method depending in any sense on immediate enumeration of all the possible readings, no matter under what search strategy, is doomed to eventual failure.

Rather, what is needed is a representation and supporting analysis system which allows us to encode what information we have as and when we receive it – hence *early* analysis – but also implicitly to encode and reason about ambiguity, so that explosions like that described above are no longer an important issue. We will need to be able to update this representation as we obtain more information from our input, possibly changing it quite a lot, maybe replacing a set with subsets, or (like Mellish's *incremental* representation and analysis system) ruling out possibilities for reference.

I will call such a representation system *adaptable*. The majority of this document will be devoted to the presentation of one such system, focussing particularly on the analysis of underspecified set reference. The goal, aside from presenting this particular set technique, is to show that any representation which is truly useful for natural language analysis will be adaptable, because, quite simply, it is never, in general, possible to say that all of the reference in an analysis has been analysed correctly, until the end of a discourse has been reached – and sometimes not even then.

A corollary of this view of language understanding is that to take full advantage of the adaptability of representations, we require the ability to represent partial sentences. Thus, local ambiguity within sentences can be represented implicitly and efficiently.

4. A Manageable Domain

Clearly, from the size of the general reference problem discussed in Section 2, I must make some kind of artificial separation between which effects we are to study and which we are not, if I am to address a manageable problem. Some researchers have chosen to limit their discussion to a particular style of language (*eg* [Mellish, 1981], which analysed A-level mechanics questions), while others have selected a very restricted world of which descriptions can be formulated (*eg* the "blocks world" of [Windograd, 1972]'s SHRDLU).

While the basic need for a manageable domain is evident, it is not clear that limiting either style or discourse domain or both maintains the full range of linguistic effects that one might wish to discuss. Certainly, the choice of A-level mechanics questions for MECHO affects the range of the analysis problems tackled by the system, to the extent that the issue of underspecified sets, for example, does not even arise.

Therefore, I propose that a better approach is to select particular linguistic features which affect reference analysis uniformly throughout linguistic usage. These may be dealt with in isolation, ideally before attempting to tackle the harder issues involved in relating reference analysis to context – *eg* focus, real world knowledge. The particular effects I propose to discuss here are those created by number, sort, and definiteness. In particular, I shall consider the effects of number and underspecification in number.

5. Definition of Terms

In this exposition, I have of necessity introduced a number of new terms. It seems that there are not enough words in the English language to have a separate term for every feature of each person's work, so some of these unfortunately overlap the terminology of other researchers. Where this is the case I have tried to make it clear whose term I am using at each ambiguous occurrence. So that the reader is forewarned, some of the most problematic of these terms are detailed below. A fuller glossary is given in Appendix A.

Adaptability

The ability of a representation to be uncommitted as to its exact interpretation, but to

change (in some well-defined meaningful way) as and when factors within the analysis require it to do so. In the system presented here, changes will always be towards less ambiguous representations.

Definiteness

The property of a noun phrase which requires that its referent be uniquely selectable from a number of possibilities. NB this is not equated here with given-ness.

Dereferencing

The process of finding and refining sets of possible referents for referring expressions (by analogy with the dereferencing process applied to variables in programming languages).

Early Evaluation

Evaluation of referential information before the traditional points (*eg* end of sentence) in sentence processing.

Given/New-ness

The property of information in discourse determining whether it is known (and so can be used for selection of candidates for reference) or new (and so should be added to the discourse representation). These two are not mutally exclusive, and only form an approximation which I will not use; I suggest that the usual equation of this property with definite/indefinite-ness is misleading.

Incremental Parsing

The incorporation of discourse information into a (partial) translation, word by word, as soon as it appears (*cf* Mellish's incremental evaluation – the notion of accumulating information in a representation).

Incremental Reference Evaluation

The piecemeal evaluation of reference, in a system where evaluation is performed early. New information may be continually added.

Introductoriness

The property of a referring expression which determines whether it causes a new entity or set of entities to be added to the world model of a discourse analysis. NB Again, this is not equated with given/new or definiteness here.

6. Summary

The material in this chapter can be summarised in three main claims.

1. We need to be able to analyse, reason about, and represent reference to sets which are subsets of other sets existing in a discourse, but not fully specified therein, in particular in terms of number.

2. We cannot, until the end of an input discourse, fix substitutions for names of discourse world entities for the referring expressions in our input, because it is not possible to be sure that a candidate entity is the correct choice until it is known that no more information is forthcoming. Thus, representations must be *adaptable*.

3. As a corollary of 2, we need to produce *partial* representations in order that we may represent intermediate stages along the path to the eventual analysis.

Most of the rest of this document is devoted to the justification of these claims. By the end of Chapter 7, I will have demonstrated how the problem discourse in example 1 can be analysed in a computational system using an adaptable representation. I will place the presentation in the context of word-by-word incremental evaluation, with as little non-determinism as possible, because this enforces an extreme requirement of adaptability on a representation, and therefore forms a stringent framework for research.

7. Afterword

This chapter has introduced the example problem discussed in this document, and a basic idea for a context in which to solve it. The rest of the document is structured as follows.

Chapter 2 is a brief survey of related work in computational analysis of reference. Chapter 3 introduces the George system, which embodies the theory presented here; also, some important basic hypotheses are presented. In Chapters 4 and 5, the George Representation Language (GRL), an adaptable discourse representation system, and the George Parser, a parser designed to produce partial representations and take advantage of the adaptability of GRL are explained and illustrated.

Chapter 6 explains how George analyses reference of the kinds covered by [Mellish, 1981], and introduces a new view of restrictive and non-restrictive noun modification. Chapter 7 extends this and shows how adaptability admits reasoning about reference to underspecified sets. Chapter 8, discusses the George system and theory and compares it with existing work. Finally, Chapter 9 outlines ideas for extension and improvement.

Chapter 2
Related Work on Noun Phrase Reference

Abstract

This chapter introduces and justifies the starting point of my approach to noun phrase reference analysis, in terms of existing work in the field. In particular, it discusses which referential features of language are to be used, and presents the contributions of three closely related pieces of work by Webber, Mellish and Haddock. The referential Principles of Steedman *et al* are also presented, since they are implicit in the work presented here. Incremental parsing is introduced.

1. Introduction

In this chapter, I will provide a context for the problem presented in Chapter 1, in terms of work by other researchers in the field of Natural Language Processing. I will focus on the work of [Mellish, 1981, *etc*], [Webber, 1979], [Haddock, 1989], and, to a lesser extent, on [Crain & Steedman, 1985], [Steedman, 1985], [Altmann, 1986], [Steedman, 1987].

Three other important pieces of work, by [Sidner, 1979], [Alshawi, 1987] and [Carter, 1988] are also concerned with resolution of anaphora. The first two mainly discuss focus, and so are orthogonal to the work presented here. The third, while having features common with the George system (in particular, what Carter calls "shallow processing") is not directly relevant to adaptability or to underspecified set reference (quantified reference is explicitly excluded from Carter's discussion).

For each of my chosen authors, I will discuss in this chapter the main issues introduced in Chapter 1, namely: reference to subsets of underspecified sets and how to represent it; the need always for *adaptable* representations of entities taking part in a discourse in general; and the need to derive correct partial representations of entities (both sets and individuals) from an input discourse. The presentation is given in the context of word-by-word incremental evaluation, with as little explicit non-determinism as possible. This enforces an extreme requirement of adaptability on a representation, and therefore forms

a stringent framework for examination of the problems and advantages of the adaptability of that representation.

Before proceeding to issues of set reference and of my representation and parsing system, I will restrict my domain of data by listing a number of linguistic issues I will not cover and explaining why. I will also list and exemplify [Steedman *et al*, 1985, *etc*]'s Principles of Reference , which will form fundamental implicit tenets of my presentation.

All of the discussion in this chapter should be viewed as providing a context for the presentation of the George Parsing and DeReferencing system, in the next five. Chapter 8 will then highlight differences and similarities between George and the other researchers' work, and the improvements embodied in it.

2. Noun Phrase Reference

Noun phrase reference evaluation is a large problem, in two respects: first, because there are so many different aspects to it, and second, because there are so many existing ways partially to solve it. Both of these respects form choice points in the search for a focus of research.

In respect of the first choice point, in order to constrain my domain to a manageable size, I must make some decisions about what I will and will not address. The broad issues I might cover here are as follows; the list is in an order which (in an informal sense) starts more in the mind of the hearer and moves "outward", as it were, deeper into the explicit discourse.

1. *Knowledge of speaker's intention*
   Knowledge of a speaker's intention in carrying on a discourse may affect a hearer's ability to analyse referring expressions (among many other things). For example, if the hearer believes (because of context, punctuation, or inflection) that the speaker is issuing an instruction and the context does not preclude this (see below), s/he is likely to assume that a sentence sounding like "Have the animals eaten?" is an order to send, say, cattle to a restaurant, and nothing to do with concern over their possible hunger.

2. *Knowledge of speaker*
   More commonly, and more obviously, knowledge of the speaker's background and likely assumptions will affect dereferencing. For example, when the speaker refers to "John", but the hearer knows two people called John, the hearer will assume that the speaker is referring to the one they both know, if s/he believes such a unique individual to exist.

3. *Inference/deduction from situation described in a discourse* (*eg* [Winograd, 1971])
   More commonly still, a hearer will make inferences, particularly regarding referential consistency of a situation described or requested. For example, if the speaker refers to "the donkey the man owns" but no man has been previously mentioned, the hearer is likely to require clarification.

4. *Inference/deduction from real world knowledge* (*eg* [Winograd, 1971])
   The hearer will not normally allow reference which would imply a situation which s/he knows is physically impossible, impracticable, or just unlikely. For example, if the speaker refers to "the donkey riding the man", the hearer is likely to require clarification, unless the discourse is presented in a context where such a situation is possible (for example, in reference to a cartoon film). Another example is *deixis*, where one refers directly to entities in the real world surrounding the discourse participants; deixis may also play a part in 3, above.

5. *Focus (and recency)* (*eg* [Grosz, 1977])
   The term "focus" has a number of definitions in the (computational) linguist's dictionary. I use it loosely here to cover various syntactic and pragmatic effects which may influence the choice between a number of possible referents for a referring expression, based on theme or thread of meaning. A good example is the idea of recency: that, given two otherwise equally likely candidates for reference, the most recently mentioned is often a better choice.

6. *Definiteness: the given/new distinction* (*eg* [Mellish, 1985], [Webber, 1979])
   The distinction between given and new information is one which has been the subject of numerous discussions in the past. In many cases, researchers have made do with an approximation to the distinction: that definite references always contain information already known, and that indefinite ones always introduce new information.

7. *Sort* (*eg* everything ever written about noun phrase reference)
   Sort information is that concerning the properties of entities existing in the world of the discourse. Sorts are often expressed as predications of entities; they may be viewed as defining properties (*eg* "a *red* ball"), or as properties arising from the situation described by a discourse (eg "a ball *I kicked*"), or as both. There is a very fuzzy dividing line between what genuinely constitutes definition and what constitutes situation information; I will address this issue briefly in Chapter 4. Sort information is a fundamental force in determining candidates for reference.

8. *Number* (*eg* [Webber, 1979])

   The number information concerning a set of individuals may in some cases be as strong as sort in determining reference. For example: "Ten Tory MPs spoke in opposition; twenty Labour Members spoke in support; the ten won the vote.". Even though the practical aspects of the situation might quite strongly suggest an opposing reading, the reference carrying nothing but number information is still unambiguous.

As did Mellish and Haddock, I will restrict the area of my discussion to points 6,7 and 8 and certain aspects of 3. This is justifiable on the grounds that the problems covered here will all be phrased in terms of reducing sets of potential referents for referring expressions (Mellish's *candidate sets*). The issues covered in 1, 2 and 5 above are all characterisable as processing operations on these candidate sets. For example, one might naïvely view the initial candidate set of a reference as being all the entities in a discourse, some of which will be ruled out as the parse proceeds incrementally. A more sophisticated stance would be that the initial candidate set is those entities which are in focus (given some suitable definition of the term). Therefore, useful work can be done by taking small discourses, in such a way that the candidate set is restricted as though by these other factors, thus keeping the problem both realistic and manageable. These processing operations probably need to be modelled in parallel with the semantic analysis and more simple reference analysis of language; the topic of a discourse, for example, will affect the discourse focus. One major advantage of this view is that it need not restrict research to a particular domain of data.

Issue 4, above, can be included on a sentential level, by the use of selectional contraints in more or less complicated ways: for example, [Wilks, 1975]'s *preference semantics* allows such decisions to be made on grounds of probabilities of correctness, or likelihoods. I will discuss in Chapter 9 how selectional constraints could be built into the theory.

In this section, I have narrowed down the area of my research into three related issues: definiteness, sort, and number. There is wide agreement among researchers that these are fundamental determining factors in noun phrase reference analysis. Therefore, it seems unnecessary to justify this choice further.

3. Steedman, Crain & Altmann: Principles of Reference

In a number of different articles, Steedman, Crain and Altmann formulate various rules or principles of referential behaviour, all of which are embodied either implicitly in the

theory presented here, or prospectively in the further stages of which it might be a front end. I will merely state and outline these principles here – they are sufficiently intuitive not to require further discussion, and are anyway more than adequately explained in the original sources.

The Principle of Referential Success [Crain & Steedman, 1986]

> "If there is a reading which succeeds in referring to an entity already established in the hearer's mental model of the domain of discourse, then it is preferred over one that does not."

For example, in 9, "a donkey" refers most readily to one of the set of ten, and not to some newly introduced donkey.

> "There are ten men and ten donkeys. Each man owns a donkey." 9

The Principle of Referential Failure [Altmann, 1986]

> "If a definite referring expression fails to restrict the set of candidate elements in the hearer's mental model of the domain of discourse to a single member, then an analysis which treats subsequent material as a modifier for that referring expression (*ie* as providing information which may lead to further restriction) will be favoured over one that does not."

For example, in 10, two noun phrases are post-modified by relative clauses. The first, marked ① is modified non-restrictively; there is only one possible referent of the noun phrase "the farmer"; thus, according to the rule above, the relative need not be treated restrictively. However, because there are <u>two</u> candidates for reference by the phrase ②, "the donkey", the precondition of the principle is fulfilled, and the rest of the noun phrase is viewed as restricting that set of candidates. Therefore, according to the rule, this relative clause must be viewed as restrictive.

> "Two donkeys lived in a field owned by a farmer. One was very greedy.
> The farmer① who was mean decided to sell the donkey② which was greedy,
> because it cost too much to feed." 10

The Principle of Referential Support [Altmann & Steedman, 1988]

> "An NP analysis which is referentially supported will be favoured over one that is not."

Here, the notion of *referential support* encapsulates the preconditions of both of the earlier rules; if an analysis is referentially supported, it contains all and only the information

required to disambiguate completely any references which are not introductions of new entities into the world model. Note that this rule does not preclude the addition of new information from referring expressions; it merely requires that the information in those expressions does not contradict that already known about the existing entities to which they are found to refer.

The Principle of Parsimony [Altmann & Steedman, 1988]

> "A reading which carries fewer unsatisfied presuppositions will be favoured over one that carries more."

This principle is like a principle of lazy evaluation: its claim is that as less reasoning is necessary in making a discourse world analysis consistent, so the human sentence processing mechanism is more likely to achieve that analysis. Consider the (very) strained anaphor in example 11a.

"Nine of the ten marbles were in the bag. It was under the couch." 11a

"Nine of the ten marbles were in the bag. The marble was under the couch." 11b

In this example, there are two readings, one of which is achieved much more readily than the other: in the "easy" reading "it" refers to the bag; in the "hard" one, "it" refers to the tenth marble. The point is that, given the ambiguity, it is extremely counter-intuitive to prefer the more obscure reading, because it presupposes the existence of an entity (the tenth marble) not explicitly mentioned in the discourse, which must therefore be inferred; thus, this reading is less parsimonious. In the easy reading, the entity already exists. In 11b, the obscure reading somehow becomes less so, because the more easily achieved reading has been removed.

The Principle of Parsimony might well be used as part of a final output stage for choosing between the various readings produced in the theory presented here. However, in the working George system all readings are produced, and it is left to some other filtering stage (or oracle) to select between them at some stage (NB not necessarily the end) in each possible parse.

4. Analysing Reference in a Discourse

4.1. Introduction

In this section, I will discuss work presented by three authors, [Webber 1979], [Mellish, 1985], and [Haddock, 1989]. For each author, I will give a brief resumé of his or her work, with particular attention to adaptability of representation and to the ability to represent and reason about sets. I will also introduce some motivation for incremental parsing and analysis of natural language.

4.2. Webber: "A Formal Approach to Discourse Anaphora"

4.2.1. Introduction

> "This thesis starts from the perspective that dealing with anaphoric language can be decomposed into two complementary tasks: (1) identifying what a text potentially makes available for anaphoric reference and (2) constraining the candidate set of a given anaphoric expression down to one possible choice." [Webber, 1979, Synopsis, page vi]

Thus, Bonnie Webber defines the context of her PhD thesis. She goes on to suggest that

> "it has only been the second task ... that has stimulated research in psychology and artificial intelligence (AI) natural language understanding".

Indeed, it would seem very hard to contradict the initial claim. Essentially, Webber is saying that, in order to refer to something, one needs something to which to refer. At first sight, this paraphrase seems almost fatuous – but Webber is quite right to make the point: in Chapter 8, I will suggest that in the past some (or even many) researchers have viewed reference as something internal to language.

The suggestion is that we need to make a clear, formal distinction between linguistic references and the entities to which they refer. Given this starting point, Webber states the main message of her PhD thesis in

> "two strong claims:
>
> 1. None of the three types of anaphoric expression that I have studied – definite anaphora, one-anaphora and verb phrase deletion – can be understood in purely linguistic terms. That is, none of them can be explained without stepping out of the conceptual model each participant is synthesising from the discourse.

2. On the other hand, if a discourse participant does not assign to each new utterance a formal representation in which, *inter alia*,
   a) quantifiers are indicated, along with their scopes;
   b) main clauses are distinguished from relative clauses and subordinate clauses;
   c) clausal subjects are separated from clausal predicates;
   then s/he will not be able to identify all of what is being made available for anaphoric reference."

She goes on to show that

> "there is an intimate connection between such a formal sentential analysis and the synthesis of an appropriate conceptual model of the discourse."

This notion of reflecting surface structure in a representation (point 2, above) is central to the work presented here. It is certain that the theory embodied in the George system would not function without the proper distinction between referring expressions and entities or without the extra-logical linguistic information encoded in the George Representation Language (which is defined in Chapter 4). Furthermore, we cannot have access to a separated structure of entities in the world model and reference to them in the discourse, unless we step outside the world model to some other level from which we can view the two – this is Webber's point 1, above.

4.2.2. Separated Representations of Reference in Discourse

Webber's central claim, then, is that we must represent the entities in the discourse world distinctly from the referring expressions which refer to them. To this end, she introduces a formalism admitting (quasi-)logical representations of the semantics of input sentences, and entities, initially separate from the formal representation, but described by an *invoking description* specified in terms of the same language. This is all presented at an abstract level, rather detached from issues of parsing and of mechanically producing the necessary translations.

Webber does not give a formal semantics for her representation language. It seems that the representation is based on first order predicate calculus, with the addition of parametric sorts (or types, in Webber's terminology), lambda (λ) abstraction, meta-variables and a definite operator, $\imath$ (*iota*). Of particular interest here are the means of representing types, sets and entities.

A type is a unary predicate, associated with a logical variable in the representation language via the quantifier binding it. Free variables are not allowed, except in special

cases, where they are used as short cuts and are not associated with any strong claims (*eg* "I" is used to denote the speaker). Examples of typed semantic expressions are as follows.

"A man wants a t-shirt." S_1: ∃(m:Man).∃(t:T-shirt).[Wants m, t]

"The man buys a blue t-shirt." S_2:∃(t:λ(u:T-shirt)[Blue u]).[Buys ιm:Man(m), t]

After Webber's semantic processing, three entities will be associated with these sentences: the man who wants a t-shirt and buys one; the prototypical t-shirt he wants to buy; and the blue t-shirt he actually does buy. These entities are defined by associated identifying descriptions, including specifications of which sentences "evoke" them, thus:

e_1: ιx: Man(x) & ∃(t:T-shirt).[Wants x, t] & ∃(t:λ(u:T-shirt)[Blue u]).[Buys m, t]
& **evoke**(S_1, e_1) & **evoke**(S_2, e_1)

e_2: ∃(x:t-shirt).∃(m:Man).[Wants m, t] & **evoke**(S_1, e_2)

e_3: ιx: T-shirt(x) & ∃(m:Man).[Buys m, t] & **evoke**(S_2, e_3)

Having deduced the existence of these entities, Webber specifies an iterative procedure by which the entities may be inserted into the original expressions to represent the translation of the sentences, to give a complete translation, in association with the identifying descriptions, like this:

"A man wants a t-shirt." S_1: ∃(m:Man).∃(t:T-shirt).[Wants m, t]

"The man buys a blue t-shirt." S_2: ∃(t:λ(u:T-shirt)[Blue u]).[Buys e_1, t]

Thus, in the final representation, Webber partly abandons her idea of a separated representation, and with it, adaptability. I will argue in Chapter 8 that this is a serious failing in the theory.

Webber uses this framework to give examples of treatments of three different kinds of anaphor: definite noun phrases, "one" anaphors, and verb-phrase ellipsis. In her treatment of definite noun phrases, she discusses set entities quite thoroughly. These set entities are the subject of the next section.

4.2.3. Set Reference, Quantification and Parametric Types

Webber uses two distinct devices to cover reference to sets in her language: the notion of a *set entity*, and that of conventional universal quantification over a type. Thus, sets can be referred to as individual entities or collectives, and explicit distributive quantification can be represented as such.

To represent sets, Webber introduces a type constructor (which she views as simply a function from types to types, rather than as a parametric type in itself) called **set**. Thus, the type of sets of men is denoted by **set**(Man). As syntactic sugar, she also introduces **maxset**, defined such that **maxset**(Type) contains just the element of **set**(Type) whose extension is largest; **maxset** might alternatively be defined logically in terms of **set**. Examples of set references and the entities they invoke are as follows. Note the ambiguity between the collective and distributive readings of the first sentence, which gives rise to two possible discourse translations. Recall that **set** and **maxset** are functions returning types, so "**maxset**(Man)m" denotes the application of a predicate name to the variable m.

1: "Some men want a t-shirt." $\exists$(m:**set**(Man)).$\exists$(t:T-shirt).[Want m, t]
"The men buy a t-shirt each." $\forall$(x$\in$ιm:**maxset**(Man)m).$\exists$(t:T-shirt).[Buy x, t]

2: "Some men want a t-shirt." $\exists$(m:**set**(Man)).$\exists$(t:T-shirt).$\forall$(x$\in$m).[Want x, t]
"The men buy a t-shirt each." $\forall$(x$\in$ιm:**maxset**(Man)m).$\exists$(t:T-shirt).[Buy x, t]

From these, Webber's entity derivation rule produces three entities, as before; except that this time there is a set of men and of non-prototypical t-shirts, and there are two possible interpretations, because of the collective/distributive ambiguity above.

1: e_1: ιx: **maxset**(Man)x & $\exists$(t:T-shirt).[Wants x, t]
& $\forall$(x$\in$m).$\exists$(t:λ(u:T-shirt)[Blue u]).[Buys m, t]
& **evoke**(S_1, e_1) & **evoke**(S_2, e_1)
e_2: $\exists$(x:t-shirt).$\exists$(m:**set**(Man)).[Wants m, t] & **evoke**(S_1, e_2)
e_3: ιx: **maxset**(T-shirt)x & $\exists$(m:**maxset**(Man)).$\forall$(u$\in$m).$\exists$(v$\in$x).[Buys u, v]
& **evoke**(S_2, e_3)

2: e_1: ιx: **maxset**(Man)x & $\exists$(t:T-shirt).$\forall$(u$\in$x).[Wants u, t]
& $\forall$(x$\in$m).$\exists$(t:λ(u:T-shirt)[Blue u]).[Buys m, t]
& **evoke**(S_1, e_1) & **evoke**(S_2, e_1)
e_2: $\exists$(x:t-shirt).$\exists$(m:**set**(Man)).$\forall$(u$\in$m).[Wants u, x] & **evoke**(S_1, e_2)
e_3: ιx: **maxset**(T-shirt)x & $\exists$(m:**maxset**(Man)).$\forall$(u$\in$m).$\exists$(v$\in$x).[Buys u, v]
& **evoke**(S_2, e_3)

When the entities are inserted as before, we are left with the following two alternative, fixed representations, given the respective descriptions, above, of the entities:

1: "Some men want a t-shirt." $\exists$(m:**set**(Man)).$\exists$(t:T-shirt).[Want m, t]
"The men buy a t-shirt each." $\forall$(x$\in e_1$).$\exists$(t:T-shirt).[Buy x, t]

2: "Some men want a t-shirt." $\exists(m{:}\textbf{set}(\text{Man})).\exists(t{:}\text{T-shirt}).\forall(x \in m).[\text{Want } x, m]$

"The men buy a t-shirt each." $\forall(x \in e_1).\exists(t \in \text{T-shirt}).[\text{Buy } x, t]$

4.2.4. Summary

In this section I have summarised the parts of Bonnie Webber's work which are relevant to the work presented here. I suggest that her basic claims are correct and important and therefore worthy of further investigation. However, there are some important issues which Webber does not discuss, the relevant ones here being reference to subsets of existing sets or particular individuals, and the status of the entities in the representation language. The former is simply not covered in Webber's examples; the latter is is undefined – in particular, it is arbitrary whether entities are individuals, sets, sets of sets, or whatever. This is a problem, for example, if we wish to refer to two existing sets with the same quantified referring expression: even if Webber defined the set union operation, which we would need (and she does not), extra type checking would always be necessary to determine the correctness of the types of the entities being combined.

In this document, I suggest that a dual-level representation is indeed needed, but that the levels need to be even more explicitly separated than in Webber's proposal, if we want an elegant and well-motivated adaptable representation of language; further, I suggest that the information contained in the separation may not in general be discarded until it is known that no further input is forthcoming, even though an intermediate unseparated (*eg* FOPC) translation may be produced. I will propose such a separated representation, covering the problems outlined above, and define an associated semantics in Chapter 4; in Chapter 5, I will specify a mechanical process for translating a subset of English into this representation; and in Chapter 7, I will define a process of reasoning about reference to subsets of sets.

4.3. Mellish: "Coping with Uncertainty: Noun Phrase Interpretation and Early Semantic Analysis"

4.3.1. Introduction

Chris Mellish's PhD thesis made a major contribution to computational linguistics in its suggestions that noun phrase reference is a process of satisfaction of constraints introduced by the global discourse, not just by the local noun phrase, and that reference analysis may be performed *early* and *incrementally*. I will discuss the advantages of early

and incremental evaluation in Section 4.5; in this section, I will outline Mellish's work, assuming the justification of his early/incremental approach – the successful operation of his system is rather dependent on its incremental nature.

Mellish's program formed a part of a much larger framework, the MECHO system (see [Bundy *et al*, 1979]). The system was designed to take as input standard A-level mechanics questions, parse and translate them, solve the problems, and output solutions. The project was successful, and benefitted from the existence of a clearly defined and fairly tightly constrained domain of data, in which many words and phrases were used in particular constrained ways.

Within MECHO, Mellish's new approach to noun phrase reference

> "involves three key ideas:
>
> 1. The representation and use of levels of meaning *between* the description provided by a noun phrase and the set of entities in a world model which the phrase is talking about. This includes the representation of partially evaluated references and various kinds of 'typical elements'.
> 2. The viewing of definite reference evaluation as a *global* problem of satisfying consistency constraints and presuppositions, rather than a *local* problem of finding an object satisfying a description. This introduces the possibility of using existing algorithms for constraint satisfaction, such as that used by Waltz [Waltz, 1972].
> 3. The idea of expressing the determination of quantifier scope and set cardinality by operations on *dependency information*, which represents the structure of sets represented by typical elements." [Mellish, 1985, p12]

Thus Mellish's work is aimed in very much the same direction as that of Webber (which was done at almost the same time), and starts from some of the same ideas: the need for a multi-level representation, and the need to encode quantification structure in the representation. Mellish does not, however, insist on the maintenance of linguistic structure found in Webber's work; in MECHO, all constraints on noun phrase reference are treated uniformly, by the same constraint satisfaction algorithm.

4.3.2. Candidate Sets and Constraint Satisfaction

One notion fundamental to Mellish's view of noun phrase analysis is that of the *candidate set*, first introduced by [Charniak, 1972]. Each referring expression is represented in the system by an entity (indefinite, reference, or set entities are available, depending on the referring expression). With each reference entity is associated a set of entities which are already known (or assumed) to exist in the discourse world, to which the referring expression (or as much of it as has been parsed, in Mellish's incremental framework) may

refer. These entities are *candidates* for reference by the referring expression; the set is the *candidate set* of the reference entity. In Mellish's system, the candidates exhibit some non-strict superset[1] of the properties required by their referring expression. As the process of analysing a particular reference in the larger sentential context proceeds, the candidate set associated with it becomes more *constrained*; so its size decreases, until, eventually, there is some number of elements (often exactly one) which is appropriate to the form of the reference (*eg* definite, plural, *etc*) – otherwise the process has failed, implying that the reference is ambiguous, or that it has no referent.

The notion of the candidate set is sufficiently general to be of use in almost any context of early and/or incremental reference analysis. It is the idea of the sets being repeatedly more *constrained* by the information contained in the noun phrases and subsequent text which allows Mellish (and Haddock) to treat noun phrase reference analysis as a constraint network problem, and to attempt to solve the sets of constraints by well-understood network consistency techniques (see [Waltz, 1972]). Haddock's writing focusses more on this aspect than Mellish's – it will be more appropriate to discuss it further in Section 4.4, with the rest of Haddock's work.

4.3.3. Set Entities and Linked Dependencies

Mellish's treatment of sets is at a less abstract, more practical level than Webber's. He extends his existing notion of indefinite entities representing individuals to set entities which represent sets. Unfortunately, at this stage, he is forced to introduce a theoretical impurity into the system, because there is no translation of the discourse as such, but only a representation of the discourse world described. As both Webber and Mellish point out, there are broadly two different ways of referring to sets in English: either as composite individual entities (predications of which may be either distributive or collective) or explicitly as sets of individuals – the difference between, for example, "The men" and "Each man". Mellish uses two different kinds of set entity to represent these two kinds of reference, because the two forms require different kinds of subsequent reference, and so much be distinguishable in a reference analysis system. Thus, one might infer that the surface form of the discourse is to some extent made manifest in the world model described by the discourse, which seems philosphically undesirable[2].

1: This relationship is significant; I will discuss it further in Section 4.4.2.

2: Although Mellish explicitly views the entities as somewhere between the language and the discourse world, he does not tell us at exactly what level they really do appear.

Mellish's constrained domain led to a considerable simplification of set reference. All of MECHO's sets were small, of definite size, and therefore explicitly enumerable – though in fact the chosen representation is much more general than mere enumeration. Mellish's main point in designing the set reference mechanism is that one often needs to treat sets as though they were defined as the cross product of other sets, and/or to generate set definitions from cross products. For example, in 12, there are six pulleys, not three.

"Two blocks, each containing three pulleys, are placed on a smooth table." 12

In fact, Mellish views sets as n-dimensional matrices, labelling each element by its position in the matrix. This decomposition into dimensions is denoted by the device of *subscripting*, where a reference is annotated with a value (which may be an uninstantiated variable) for each dimension to yield a reference to some subset. Thus, individuals can be picked out by supplying a numerical label for each dimension; and subsets containing all the elements along one or more particular dimension(s) can be selected simply by leaving the particular label associated with the relevant dimension(s) uninstantiated. Mellish uses a neat positional encoding in the representation of these *dependencies* to associate each label directly with the correct dimension. In order to constrain the domain still further, he makes a simplifying assumption concerning set relationships such as that expressed in 13; he assumes them to be one-to-one correspondences, which he represents by unifying the subscripts of the related set entities, to create *linked dependencies*.

"The particles are attached to the strings." 13

This was certainly a reasonable decision in context, but in the more general case leaves a multiplicity of readings unexplained. The automatic generation of representations of these readings is one goal of the work presented here.

Finally, it is not clear how (if) Mellish deals with problems like those presented in Chapter 1, where the cardinality of a set is unknown – indeed, for purposes of decomposition, through the subscripting described above, he requires that it be known.

4.3.4. Summary

In this section, I have introduced those parts of Mellish's work which are relevant to the theory presented in this document. In particular, I have suggested that, like Webber, Mellish exhibits a certain disregard for purity of representation, and I have pointed out an

area deliberately excluded by Mellish which deserves further investigation – that of reference to subsets of underspecified sets. In Chapter 4 and 6, I will introduce a fully separated representation for natural language utterances; in Chapter 6, I will specify a reference analysis system which allows external constraints arising from other (maybe subsequent) noun phrases to affect each local noun phrase reference evaluation, which is then generalisable to general discourse world inference within the existing program, as in MECHO. In Chapter 7, I will specify a reasoning system which aims to subsume Mellish's useful ideas of subscripting and dependency linking and to generalise them to allow generation of the readings available from references to underspecified sets and their subsets.

4.4. Haddock: "Incremental Semantics and Interactive Syntactic Processing"

4.4.1. Introduction

In [Haddock 1989], Nick Haddock's general aim is to investigate the idea of carrying incremental parsing and the interaction of syntax and semantics (and reference) rather further than [Mellish, 1981], in that his system composes the translations of noun phrases incrementally, from words, rather than composing semantics incrementally from noun phrases *et al*, as does Mellish's. Thus, Haddock's evaluation is both early and incremental, but is often more so than Mellish's.

Primarily, there are two sections to the work; one of these deals in depth with issues of complexity in network constraint satisfaction, most of which, though interesting, is not relevant here. The other, which is directly relevant to my work, concerns issues like those considered by [Winograd, 1972], where definite references are mutually dependent. Haddock's proposal is that

> "a definite NP should refer uniquely by the time it is syntactically closed".

For example, in a situation where there are two hats and two rabbits, and one of the rabbits is in one of the hats, the noun phrase in 14 is unambiguous, even though the two noun phrases of which it is composed are ambiguous in isolation.

"The rabbit in the hat" 14

This might be thought to present a problem to an incremental analysis, since the first embedded NP is ambiguous until the meaning of the word "hat" is incorporated into the semantics. However, Haddock uses the fact to distinguish between the restrictive and

non-restrictive readings of the prepositional phrase. In the non-restrictive reading, the prepositional phrase is viewed as associated with the head noun phrase; in the restrictive, it is associated with the noun. In this situation, where there are two rabbits, the head noun phrase in isolation does not have a unique referent – therefore, under Haddock's proposed rule, the non-restrictive reading is deemed to have failed, and the restrictive (which passes the test) is presented as the correct analysis.

Mellish's approach to this issue is rather more liberal – no enforcement of such a rule is attempted. Now, Haddock suggests that his rule encapsulates the force of [Altmann, 1986]'s Principle of Referential Failure. In fact, this is not so: Altmann's rule states that when a definite referring expression does not have a unique referrent, an analysis of subsequent text which has the effect of modifying the referring expression will be preferred over one that does not. Haddock's rule requires that the immediately subsequent text be regarded as modifying, which is a much more (too?) strict requirement.

4.4.2. Noun Phrase Reference Analysis as Constraint Satisfaction

Both Haddock and Mellish use the idea of constraint network satisfaction as a means of expressing and refining the relationship between a referring expression and the members of its associated candidate set. Haddock's work is partly directed at a deeper understanding of the issues of complexity involved in this, which I will not discuss here (though it is worth mentioning that there are operations in my system which constitute adaptations of Haddock's constraint network system – the exact relationship will be pointed out in Chapter 6).

Constraint satisfaction, or some equivalent of it, is likely to be fundamental to any incremental attempt to analyse reference. There is an important point, though, which neither Mellish nor Haddock explicitly addresses: the issue of what constraints to represent, and how to use them. [Charniak, 1972] introduces the distinction between *overspecified, underspecified* (meaning "noun phrases containing new information" and "noun phrases with a non-empty, non-singleton candidate set" respectively) and ordinary noun phrases in the context of definites. Haddock's formalism explicitly excludes these: his referring noun phrases always contain only given information, and must have singleton candidate sets by the time they are syntactically closed. Mellish's domain of data deliberately excludes Charniak's "overspecified" noun phrases, and the "underspecified" ones, which anyway hardly ever arise, again because of the style of language covered, are allowed to proceed under the implicit assumption that ambiguity will be resolved before the end of the input discourse.

Another distinction raised by Charniak is that of sort comparison by necessity or possibility. His point is that entities existing in a discourse world are available for reference by a noun phrase not only if they are known to have the properties required by the noun phrase, but also if they are not known not to have them. Thus, if we have exactly one ball in our discourse world, and we parse the phrase "The red ball", it is incorrect to rule out the existing ball as a candidate because we do not know it is red. Indeed, we might want to add the new property to our representation of the ball (maybe with an indication of its special source, to allow revocation later). Charniak calls this style of candidate selection the "double negation technique"; in terms of candidate sets, it means taking the set of candidates which do not contradict the properties in the noun phrase, rather than just those that explicitly exhibit them. Neither Mellish's nor Haddock's theory takes this view, and so both are unable to deal with the simple "ball" example above. In fact, this is the closed/open world distinction. Mellish and Haddock both use a *closed world assumption* – that is to say, nothing is true unless it is provably so. Charniak's *open world assumption* is the converse of this – anything may be true unless it is known to be false. This distinction will be important later in the discussion of George.

4.4.3. Summary

In this section, I have summarised those aspects of Haddock's thesis directly pertinent to the work presented in this document. In Chapter 8, I will raise a problem with his theory of reference, in the notion of the closure constraint; since he does not cover plurals, the problem of set reference does not arise.

In Chapter 6, I will propose a theory of reference which subsumes Haddock's (NB while making his complexity arguments none the less valuable). Interestingly, this subsumption arises by relaxing Haddock's definition, rather than tightening it, and using the implication of [Crain & Steedman, 1985]'s Principle of Referential Success, that a speaker is generally intending to be comprehensible, which seems not unreasonable. This assumption will lead us, in general, to assume an open world instead of a closed one when analysing reference.

4.5. Early and Incremental Evaluation of Reference

4.5.1. Introduction

The main issue under consideration in [Mellish, 1981] is that of *early, incremental evaluation of noun phrase reference* – that is to say, the attempt to analyse the reference of

noun phrases as or soon after each phrase arrives at the input of a parser, and not at some later stage, when the entire sentence has been parsed. In this respect, Mellish's work is seminal – prior attempts to deal with the reference problem had ignored or postponed this "procedural" aspect of language understanding in favour of more "declarative" views based on complete sentences (though [Bobrow & Webber, 1980]'s contemporary work on incremental description refinement used the incremental approach on sense-semantic refinement; and Winograd's SHRDLU used local information, performing analysis at the end of each noun phrase).

In Mellish's terms, *early evaluation* means that attempts are made to analyse reference as or soon after any disambiguating information is received; *incremental evaluation* is the ability repeatedly to incorporate new constraint information into a representation so that evaluation is in some sense a continuous process. Some form of incremental analysis is a precondition of early evaluation, since in general it will always be necessary to add constraints to an existing representation – *eg* in processing a definite noun phrase containing an adjective (the initial constraint) and a head noun (the incremental constraint).

[Haddock, 1989] extends the idea to a finer granularity – Mellish did not insist on incremental analysis of the words within a noun phrase, but only of noun phrases with respect to the overall structure of the discourse. Haddock suggests that a word-by-word incremental analysis can be useful in disambiguating some perennially problematic ambiguous constructions.

Haddock's usage of the term *incremental*, then, conflates both of Mellish's terms, since his evaluation is maximally early (within the bounds of his parsing strategy), and is incremental word-by-word, at least within noun phrases. Again, Haddock's view of incremental evaluation is of continual addition to sets of constraints.

Why, then, is there such interest in early and incremental evaluation, and why is the work presented here placed in that context?

4.5.2. Some Evidence for Early/Incremental Evaluation

The argument for early evaluation arises mostly from psycholinguistic sources, though there are some practical computational considerations. To start with, the intuition in many (most?) people that noun phrase evaluation is performed, at least to some extent, word by word is so strong as to be unquestionable.

More formally, the various experimental results reported in [Marslen-Wilson, 1975], [Marslen-Wilson & Tyler, 1980], [Marslen-Wilson, 1987] and [Shillcock, 1982], strongly suggest an early/incremental aspect to language understanding. [Altmann, 1987]'s results imply an earliness in human language processing, at a sentential level, on account of the existence of so-called *garden path* sentences. Garden pathing is an effect produced by syntactic ambiguity in a sentence, where one reading is (in some sense) much less likely than another. If the ambiguity is resolved late on in the parse, and the unlikely choice is found to be correct, the reader's attempt at parsing is detrimentally affected, and can sometimes fail altogether. An example of garden pathing is given in sentence 15, where the effect is due to lexical ambiguity in the word "raced", between active verb (the favoured reading) and passive participle. This reduced ability to find alternative solutions points to some commitment by the human sentence processing mechanism to a particular reading before the disambiguating word(s) is(are) read; thus, analysis proceeds early and incrementally (but not adaptably), and the desire to find the first possible analysis leads to the parsing failure.

"The horse raced past the barn fell." 15

Another issue related to early evaluation is raised in [Ritchie, 1976]. Like Mellish, Ritchie suggests that global factors, external to any individual definite noun phrase, affect reference evaluation and that analysis must proceed beyond the end of the phrase. One of his examples is reproduced in 16.

"The President of the United States signed the test-ban treaty in 1962." 16

Ritchie's point is that we cannot ascertain the individual referred to by the subject noun phrase until we have parsed the entire sentence, and know the date associated with the context; thus, a global constraint is being applied. In similar vein, [Mellish, 1985, p5] suggests that

> "The notion of a *referent* is something shaped by the context of a noun phrase's use, rather than a simple function of the noun phrase itself.".

Ritchie's example, in 16, might be taken as a piece of evidence for that claim. [Haddock, 1989], though, contradicts this view, on the grounds that the presidential reference in 16 is in some sense intensional, and does not refer to a particular individual, but to a prototype – whoever fills the US presidential function at the appropriate time. It is harder to argue this position, though, if the noun phrase were, say, "John's newest child", where the reference is much less obviously prototypical. Haddock makes this point

to help justify his main theory (see Section 4.4). I will argue in Chapter 8, however, that even if Haddock's dismissive response to Ritchie's example (and to a similar one raised by Mellish) is correct, the conclusion based on it is drawn from incomplete data, and is seriously flawed.

One practical advantage of early reference processing is that it can actively aid disambiguation, in cases, for example, where an entity in the discourse is a candidate for reference by a pronoun in the sentence which introduced it. Consider examples 17a and b.

"A man feeds a mare which he owns." 17a

"He feeds a mare which a man owns." 17b

In case a, "he" must refer to the entity introduced by "A man", or to an entity already introduced by prior discourse. In case b, "He" can never refer to the man entity introduced in the subordinate clause, but must always refer to some existing entity. There is an open question as to the conditions under which such an initial pronoun may be introductory – a reasonable first approximation might be that it is only introductory if there are no entities to which it might refer (*eg* if this is the first sentence of a new discourse in a new context, or if all the known entities are female). The point is that, with one well-defined exception, which I explain below, an introductory phrase must precede a pronoun which refers to it – which indeed seems obvious from the terminology. However, unless our analysis is in some way word-by-word (and in this I include, for example, time-stamping the words as they are received, and then processing noun phrases in that order at the end of the sentence), the man introduced by the translation of 17b will need to be ruled out as a candidate for reference of the pronoun by an explicit rule, which seems unnecessary.

The exception to this referential rule is where the pronoun occurs in a prefixed subordinate clause. [Haddock, 1989, pp 2-3] merely dismisses this as sufficiently rare to be dealt with effectively as a special case. I suggest that a better view is to ensure that one's representation makes the distinction between this and other cases clear, by including subordinate structure in it, as does Webber's; then, detecting a special case is more clearly a viable proposition. The representation I will propose here does exhibit this distinction. However, for the purposes of this document, I shall follow Haddock's line, and discuss this aspect of the issue no further detail.

4.5.3. More Directly Relevant Advantages

These, then, are some justifications for investigating further the viability of incremental parsing in general, and early noun phrase reference analysis in particular. There are,

however, two properties of a reference analysis system whose parsing and evaluation are strictly incremental, and so maximally early, which are ideal for my purposes here.

The first is that the efficiency of an such implementation can easily be greatly enhanced by sensible use of parallelism: in particular, repeated evaluation of intermediate results can be reduced dramatically, if not removed altogether. This is certainly possible in non-incremental and/or backtracking systems, but it is much harder. Since one of the arguments against incremental semantic analysis is the overhead involved in cases where re-evaluation is necessary (*eg* in a backtracking parser), this is a significant point. In Chapter 5, I will present the first steps towards a parser which will take maximum advantage of this parallelism, through use of an ambiguous adaptable representation.

The second property, more important here, is that a strictly incremental system provides the most extremely (within the bounds of reason) stringent framework possible for the development of a theory of adaptable representation of discourse reference, as was introduced in Chapter 1.

Finally on this issue, [Crain & Steedman, 1985] suggest that some kind of preference analysis between readings is required, and that the corollary of this is the requirement for a parser which produces possible readings word by word, in parallel.

5. Summary

In this section, I have discussed the relevant work of those researchers whose theories form the foundation of the work presented here (*viz* Webber, Mellish, and Steedman *et al*), and also a theory developed concurrently in a similar context – that of Haddock.

I have broadly suggested that the ideas of Webber and Mellish are complementary, and that both may be placed in a context of strictly incremental parsing. I have also suggested that Haddock's contribution lies mostly in his analysis of parsing and network complexity, because a fundamental part of his theory of reference (closure constraints on singular definite noun phrases) is flawed; I will substantiate this in Chapter 8.

In surveying this selection of existing work, I have highlighted several common points of interest, and suggested that the theories might be improved in some way. In particular, I have emphasised the issue of representation, and of separation between discourses and worlds described by them, which, even if it does not lead to a better theory, is necessary for philosophical purity. I have suggested the need for *adaptability* in representation, on the grounds that only when a discourse is known to be finished can one irrevocably

amalgamate the two levels of the separated representation; in George, amalgamation will never take place, on grounds of theoretical purity.

I have pointed out (as have others, no doubt) that one must take great care in choosing a restricted domain of data, when attempting to develop a general theory: Mellish's choice, for example, leads to one non-general (but true) conclusion, but is justified in context; Haddock's restriction to single sentences, with noun phrases containing only "given" information, is not (I will argue in Chapter 8), because it leads to a conclusion which is actually false.

6. Afterword

This completes my discussion of related work. The next five chapters introduce the George system and its associated theory of reference. Chapter 3 is a general overview, which presents some fundamental principles and gives a feel for the operation of the working system. Chapter 4 introduces the George Representation Language and its semantics. Chapter 5 presents the George Parser. Chapter 6 explains how the George DeReferencer analyses noun phrase reference, while Chapter 7 shows how to generate readings of sentences containing incompletely specified set references. Chapter 8 will discuss the relationships between the theory embodied in George and the earlier work of the other researchers presented in the foregoing sections.

Chapter 3
The George Parsing and DeReferencing System

Abstract

An overview of the George system is presented, forming an introduction to the subsequent four chapters. The main facets of the George system are introduced, along with some of the motivation for the decisions taken and assumptions made in their design.

1. Introduction

The George Parsing and DeReferencing System is an attempt to synthesise some generally prevalent and some rather more idiosyncratic ideas into a uniform system for translating a subset of English into a more computer-friendly form. It does so in such a way as to present a solution to the problem of set reference detailed in Chapter 1.

The particular subset of English which George covers has been deliberately chosen to be independent of stylistic context. By this, I mean that it is not limited to a particular usage of English, as was, for example, MECHO [Bundy et al, 1979] (see Chapter 2).

The especial focus of the work presented here is one particular aspect of noun phrase reference evaluation. For this reason, the design of all aspects of the George System relating to other linguistic and computational issues has been based on well-tried existing approaches. For example, the George Representation Language borrows a number of useful ideas from First Order Predicate Calculus; the George Parser uses a variant on a standard grammar formalism.

It is important to understand that it is the combination of the various aspects of the George System, and not any individual feature in isolation, which enables it to fulfil its function. This fact must be borne in mind while studying this and the subsequent four chapters, which cover the three main sections of George in detail.

2. Fundamental Hypotheses

The George System is based on some fundamental theoretical hypotheses which are in themselves worthy of more thorough linguistic and/or philosophical research. George does not attempt to justify these hypotheses in abstract, but it does show that they can be useful in performing the task for which George was designed. The hypotheses are not the central focus of the work presented here, but are sufficiently important to merit theoretical explanation before George is discussed in detail.

2.1. Referring Expressions and Existentials

One such hypothesis concerns the differing status of existentials in natural language and in logic: which can be encapsulated in the question of whether one can refer to an entity which does not exist in the "real world", and how one is to represent such a reference if it can legitimately be made. It is sometimes claimed that the means by which we refer to unicorns is in some essential sense different from the way we refer to, say, horses, simply because horses actually exist in our environment and unicorns do not.

I claim that it is permissible, and, in fact, useful, to suppose that any entities may be referred to in a discourse in a uniform way, irrespective of whether they exist in the real world or not, as long as all participants in that discourse – that is, both speaker(s) and hearer(s) – already have, or are provided with, some personal definition of what that entity is. Further, this is so even if these personal definitions are not consistent between the participants. For example, as long as all participants are aware that unicorns have horns, they can talk about someone being run through by one, and neither the real existence or otherwise of unicorns nor non-relevant differences in their definition makes any difference at all to the information content of the discourse in abstract. We run into trouble only when (if) we try to map the entities in such a discourse explicitly into the real world (and even then, only if we require proof by direct evidence) as it is perceived (at or prior to discourse time) by the participants, or into a discourse world in which the entities in questions are denied existence. This is so because this issue of the existential status of discourse participants does not present a problem until it makes, for example, a claim disprovable or an imperative impossible to perform with respect to the discourse world, or with respect to the real world which the dicsourse world claims to model.

To take this argument one stage further: it is perfectly possible for a number of people to carry on a discourse referring (with identical referring expressions) to objects which are different for each of them – consider, for example, a story about John, when the speaker

means John Smith, and the hearer is thinking of John Brown. There is nothing wrong with the formation of the discourse *per se*: it is the speaker's attempt to convey certain information which has failed.

Similarly and finally, it does not make any difference, nor, arguably, does it matter in a more general sense, if the speaker, and not the hearer, believes that unicorns have trunks, tusks, large ears, and no horns. This is so, again, only as long as the speaker does not explicitly refer to these properties and thus contradict the information the hearer is using to analyse the referring expressions. Nor may the speaker attempt to extend the reference into the real world, because the hearer will then be presented with a corresponding inconsistency from a different (perhaps visual) source, and the discourse will again be shown to have failed.

From this argument, it would seem clear that a discourse, however represented, has a status in some sense abstracted from the world with respect to which it is to be interpreted (which, of course, may or may not coincide with the real world). If this is so, we must suppose the existence of a mapping between whatever represents the entities taking part in the discourse and their correspondents in this discourse world. Informally, we might say that this is the mapping between what a speaker or hearer understands by a referring expression and referent to which it "really" refers in the discourse world. It follows that this mapping must be dependent upon (and, sometimes, known only to) the agent creating and using it, so that a number of people may discuss different entities while appearing to discuss the same one, as in the "two Johns" example above. The mapping must also allow the existence of "discourse world entities" which do not correspond with any "real world entity", as for the unicorn, above. Similarly, to allow for the potential difference in interpretation of referring expressions themselves by different speakers and hearers even when given identical discourse worlds, we must acknowledge a similar mapping between referring expressions and the "understood" entities in the discourse world with which they are associated by the discourse participants.

George's reference analysis system is built on this assumption. It supposes a three-layer representation of a discourse, which is then related by a further layer to the real world. The layers (ordered by increasing separation from the linguistic material itself) are: the translation at the linguistic level, including, for example, the sort and definiteness information in a noun phrase; the hearer's discourse world level, representing the entities themselves as understood by him/her to be defined by the discourse; the discourse world; and the real world level. There is a fully defined mapping between the first two (by definition – referring expressions give rise to the understood entities) and possibly partial mappings between the others.

When we are considering the analysis of noun phrases in a discourse, we can discuss these three mappings separately, or, indeed, bypass two of them altogether. This is the course I shall take with this discussion of George: the mappings from understood world entities to discourse world entities, and thence to real world entities will not be considered, because, as suggested above, it does not affect the behaviour of the analysis of the discourse at the level in which I am interested here.

I will explain in Chapter 8 that split level representation and the necessarily associated inter-level mapping are also necessary for adaptability. The levels of representation proposed here are summarised and placed alongside their George analogues in Figure 2c.

2.2. The Representation of Discourse Surface Form

It will become clear later that the hypothesis in 2.1 has a bearing on several aspects of the design of George's style of representation. In particular, not surprisingly, it affects the way I represent referring expressions and their referents.

To support the mapping outlined above as part of our reference mechanism, we must at all times hold the information from referring expressions in a form in which it is identifiable as such as well as separate deduced or *a priori* knowledge about referents. This is also a requirement of adaptability – we must know where information has come from if we are to manipulate it in full generality. One way of viewing this constraint is to say that we will represent referring expressions at a level rather closer to its surface form than that of the discourse world or that of the conventional view of semantics. In particular, the obvious thing to do is to represent the referring expressions. This is preferable to the only alternative course of calculating some information of intermediate status, which entails unnecessary extra work (both for the system and its implementer), and which would make the theory more complicated than necessary.

Having taken this first step towards a representation of referring expressions which is closer to the surface form utterance than the "deeper" representations, closer to semantics, to which we have become accustomed in much of the literature, we will begin to see that there are often advantages in applying the same approach to other aspects of discourse. For example, [Webber, 1979] has shown that it is often useful and sometimes vital to maintain some representation of the surface structure of relative clauses when attempting to deal with noun phrase reference.

I must emphasise that, while it might well be useful to specify mappings between "surface form" expressions and corresponding representations nearer the discourse world level, it

is nevertheless not the case that this surface form may be converted into a pure semantic representation (as opposed to one containing surface form information) before the absolute end of the discourse. This is because it is not possible to say that reference has been evaluated correctly until no more information is forthcoming (or even then, in some cases!). Therefore, until this point, the representation must be adaptable in some sense; therefore, we need all the information we can get in order to recalculate correctly the references which may need to be adapted.

It will become clear in this and the succeeding chapters that this "surface form" hypothesis affects the design of George in a number of ways. The most obvious effects appear in the design of the representation language; this, in turn, though, has implications throughout the rest of the system, both in terms of theory and implementation.

2.3. The Representation of Ambiguous Sentences

The last of my three hypotheses follows in part from those introduced above. It concerns both computational efficiency and ideological cleanliness. It also gives George the property of adaptability in a strong sense.

The assumption is that, wherever possible, it is better to represent ambiguity by means of a single genuinely ambiguous representation than by more than one unambiguous one. The reasoning behind, and the implications of, this depend on the particular kind of ambiguity in question.

First, if ambiguity resides in the question of which of a number of entities is the referent of a particular referring expression, then it is pointless to maintain two full translations of the discourse to represent it, in view of the foregoing comments about surface forms and referring expression/referent mappings. Instead, the ambiguity could easily be represented by allowing the reference mapping to be one-to-many instead of one-to-one. This is the candidate set approach of [Charniak, 1972] and [Mellish, 1981]. Conveniently, this also allows us, if a supposed referent suddenly proves to be unsuitable to a referring expression, simply to remove it from the mapping. If all the potential referents for a referring expression disappear in this way, we can simply insert a new one, on the assumption that the reference is therefore introductory.

Less obviously, if ambiguity resides in the syntactic category of a word, in the vast majority of which cases it will be resolved one or two words later, it is pointless to create new versions of the translations of the discourse only to throw them away again almost immediately. For example, consider the various usages of the word "is" – as an auxiliary

verb, as an auxiliary to an auxiliary verb (*etc*), as a copula, or as an equative. This could give rise to a number of possible translations for any one sentence, the exact behaviour depending, of course, on the parsing strategy being used. It is therefore much more elegant to use an intermediate representation which can subsequently interact with the translations of succeeding words to choose the correct meaning without all the potential computational effort of creating new translations; in this way, we can gain at a computational level, in being able to parse our input more efficiently. This, again, is an example of a practical advantage of adaptability.

3. A General Overview of the George System

3.1. Introduction

Having discussed these fundamental ideas, it it now appropriate to introduce the George Parsing and DeReferencing System, which is the embodiment of the theoretical approach under discussion here. The system described is a direct implementation (or, sometimes, simulation, because of practical limitations) of the theory presented in this and the succeeding chapters.

At the highest level (*ie* from the point of view of the user) the George System behaves as a "black box", taking typed input in a small subset of English, and delivering output which I claim is a representation of the surface form (and maybe some inferred information) and reference of that input in a form more easily manipulable by a computer.

The input coverage of the current implementation is small, because neither the grammatical nor the lexical variation available to the system need be very great to recreate the particular problems of reference with which we are attempting to deal here. Therefore, it is useful to limit the size of the lexicon, because this increases the running speed of the system, making it more efficient for experiment. Limiting the grammatical coverage available (within reason), helps ensure that issues do not become clouded, because they can be considered, as it were, in isolation.

The output of the George System is expressed in the George Representation Language, a logic-like representation based loosely on the conventional Predicate Calculus. This representation expresses translations of the input linguistic utterences in a form very much closer to their original surface form than many representations used in comparable work, which usually claim to represent a "deeper" form – the semantics of the discourse. The theoretical reasons for this have already been introduced.

In the event that a discourse input to George is unresolvably ambiguous and the ambiguity cannot be represented by a single adaptable expression, more than one translation is produced (in parallel) at the output.

Importantly, the George Representation Language may be translated into the more conventional formalism of First Order Predicate Calculus. Not only does this make the translations George produces less inscrutable to the user, but it also supplies a semantics for the Representation Language *per se*. It is important to understand that this translation involves loss of some surface form information.

3.2. Some Terms

It is now appropriate to introduce some of the terms which I will to use to discuss reference throughout the rest of this document. It is regrettable that some of the George terminology is at odds, sometimes in confusing ways, with more common linguistic usage. This is because there are no other suitable words for the concepts we need to express here. Appendix A is a Glossary of these and other terms.

Reference

The level of representation of referring expressions nearest the input utterance is the *reference*. In George, the reference representing any particular noun phrase is a translation of the information contained in that noun phrase and sometimes some information inferred from it by the system. It does not represent the entity to which reference is being made – *ie* it is not the same as the reference of a noun phrase in the common linguistic sense, in that it does not represent a particular entity or set of entities. It is in fact, like anything represented in the George Representation Language, a (partial) representation of the utterance, closely related with its surface form – which, as [Webber, 1979] shows, is useful in the analysis of noun phrase reference. Thus, it is only indirectly associated with the set of entities to which reference is being made. It may be helpful to view the information carried by a George reference as a set of constraints which is (by definition) a subset of the total set of constraints defining the item in the discourse to which reference is being made.

Entity Token

The next level of representation of referring expressions, in the mind, as it were, of the hearer, and one step away from the surface form of the discourse, is that of the *entity token*. The entity token is the point at which the (constraint) information gleaned from the various references within a discourse to a particular entity collects. As such it may be viewed as something like the sense of the entity or entities to which reference is

being made. We may treat these entity tokens as *specifying* a set of entities which are candidates for reference by the references associated with them by *binding* (*qv* below). The entity tokens involved in a discourse are stored by the George system in the database labelled "Entity Tokens" in figure 2a. They are thus conceptually separate from the discourse, which position was defended in Section 2. The connection between references and entity tokens is made by the bindings shown in figure 2a. These are inferred by the DeReferencer, and are best viewed simply as a mapping between the set of references in the current discourse and the set of corresponding entity tokens.

Discourse Entity

The final level of representation of discourse is that of the *discourse world*, which contains *discourse entities*. This is the level at which we represent the actual things in the discourse world. It will not be nccessary to discuss discourse entities in more detail here, since references, bindings and entity tokens will be enough for our purposes.

Binding

The three levels of representation introduced above are no use in isolation. In order that we may express the relationships between them, I will introduce ths ideas of *binding* and *specification*. The latter is not important here, because it forms the linkage between entity tokens and (sets of) discourse entities, which we are not discussing, and is included only for completness. Bindings, however, are relevant. A binding is a relation between a reference and a set of entity tokens. It represents, in informal terms, the relationship between a referring expression and a token *specifying* its referent. Specification represents the relation between that token and the referent itself.

Bounding Constraint

A Bounding Constraint is an arithmetic equation between the upper bounds of set sizes. The Bounding Constraints are stored alongside the bindings; they are fundamental to the adaptability of the system.

3.3. The basic George System

Within this high-level unit, there is a number (greater than zero, but always small) of *basic George systems*, each of which receives its own copy of the English input as analysis proceeds. Each system notionally contains its own complete representation of the discourse so far, its own Parser, and its own DeReferencer.

Each one of the basic systems represents one or more possible translation(s) of a sentence. In general, spawning of new basic systems occurs when the addition of a new word to the

current sentence yields two or more (partial) translations with syntactic categories so different as to be unrepresentable by single ambiguous adaptable representations. For example, the combination of the translation of a verb may be either in- or mono-transitive with that of a subject noun phrase, as in "A man eats..." – conventionally, this yields (partial) translations with category s and s/np respectively, which are not, in George, representable by a single ambiguous adaptable translation (though there is no reason in principle why they should not be so).

This basic George system, then, consists, theoretically, of two computational processes (Parser and DeReferencer) operating in parallel over a shared data set (Discourse Memory, Entity List, and Bindings) and communicating data between themselves and that data set. The concept of success and failure familiar to Prolog programmers also applies: it is necessary for both the Parser and the DeReferencer to *succeed* (according to criteria specified later) for a particular translation to be proposed as correct.

Now, in anything other than a trivially simple discourse, this replication in parallel of the translations of a sentence which cannot be bundled with one or more others might appear to present a potential danger – of a combinatorial explosion of structures and processes. However, because of George's fundamental assumption that ambiguity appearing in the input data should be adaptably maintained as far as possible into the discourse, there is little danger of a damagingly large amount of non-determinism arising. This assumption effectively defers the explosion problem until either we are queried in a way which insists on the resolution of ambiguity (in which case, if we do not have an unambiguous translation, we are entitled to ask for more information) or the ambiguity disappears as a result of further input.

The structure of the basic George system is represented in figure 2a (2b is a key to the symbols used). A snapshot of the whole system translating an ambiguous discourse could be represented as a number of such diagrams, stacked up, as it were, all connected to the same data input. Figure 2c shows the relation between the levels of representation proposed in Section 2 and the components of George.

The top left and right boxes in figure 2a represent the George Parser and DeReferencer, respectively. As may be seen from the diagram, the Parser receives data input from outside the system, and sends output to the discourse memory and to the DeReferencer.

The data flow from the Parser to the Discourse Memory is in the form of sentences in the George Representation Language (GRL). That from the Parser to the DeReferencer is in the form of sets of fragments of GRL, extracted from the full GRL sentences by the Parser. These represent the surface form of referring expressions; they are effectively ordered by time of arrival at the input, by virtue of the parallelism between the Parser and the

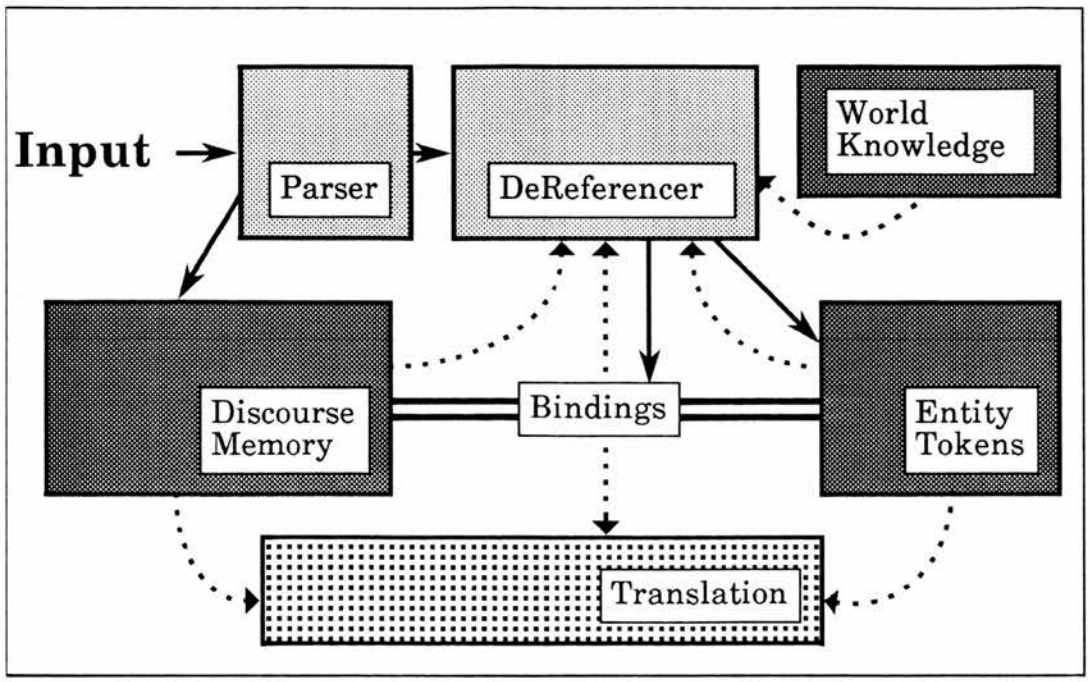


Figure 2a: *Functional Layout of George*

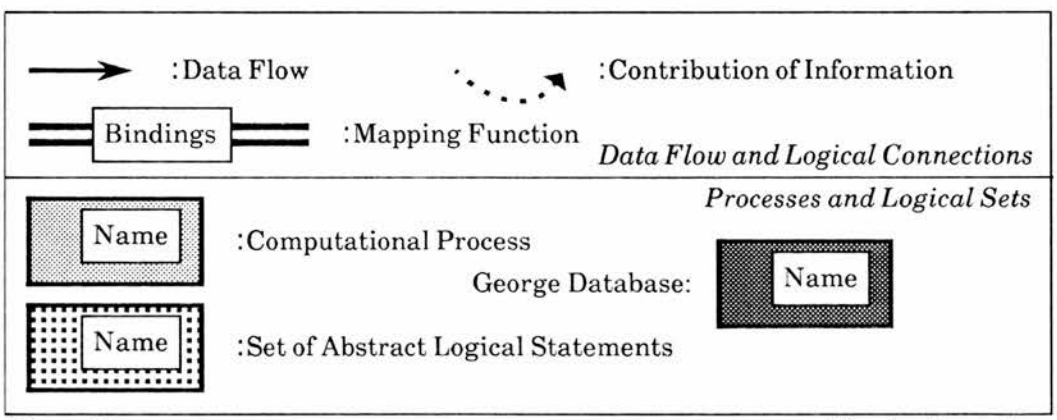


Figure 2b: *Key for figure 2a*

DeReferencer. Each time it receives a new fragment, the DeReferencer checks whether the referring expression is new or an updated version of an existing one. If the latter, the old representation is replaced with the new. The DeReferencer places new tokens representing total knowledge about the referents in the discourse in the Entity Tokens database, or updates information about those already there. It also places ordered pairs of representations of referring expressions and sets of representations of referents in the Bindings database.

The Discourse Memory database contains the surface form translations, expressed in the George Representation Language, of each sentence in the discourse so far, in order.

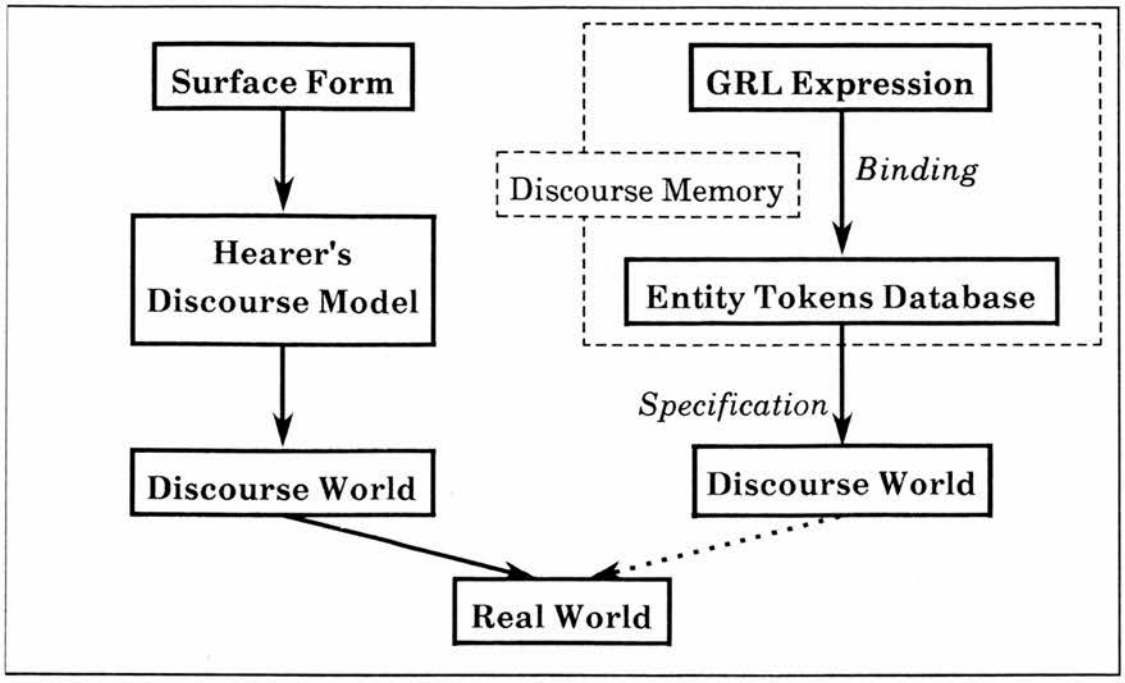


Figure 2c: *Levels of Discourse Representation and their George analogues*

(Recall that there is a separate Discourse Memory in each of the basic systems representing a reading of the discourse, where those readings cannot be expressed by ambiguous notation.) It combines with the information contained in the Bindings and Entity Tokens databases to yield a translation of the sense of the discourse. Note that sentences (or rather, syntactic entities which are deemed to be self-contained, such as noun phrases) only appear in the Discourse Memory when they have been successfully parsed in entirety. In Chapters 4, 6 and 7, I will show how sentences in the Discourse Memory may be manipulated to allow reasoning about quantified sets.

The translation, then, of George's input is contained in these three databases. It is possible to convert this tri-partite representation at any time to a more conventional form (FOPC – see Chapter 4); but it is really only useful to do so when the discourse can be said to have reached a "steady state". This is because of the issue of uncertainty mentioned before – we cannot throw away any discourse information until we know we have a correct analysis, and we cannot know that until the discourse is finished.

It may seem a little strange at first sight that the Parser takes no data input from the rest of the system, since this might be expected to preclude failure of a translation on the basis of spurious or incomplete reference. In fact this is not the case, because of the requirement that both the Parser and the DeReferencer must succeed in a particular translation for their output to pass into the data structures over which they work.

The DeReferencer takes its input from the Parser and compares it with knowledge already recorded in the lists of bindings and entity tokens (defined in full in Chapter 4). On the basis of the result of this comparison, it decides whether the input is referentially coherent, and makes additions to George's data structures representing the translation. The DeReferencer knows nothing at all of the semantics of the sentences parsed by the system, other than the information contained in the noun phrases (including relative and prepositional post-modifiers – the only predicates passed to the DeReferencer appear in these) and a very limited idea of the surface structure (*viz* subordinate structure). All it can do with this information is a simple pattern match to ascertain the reference of *context extended references* (also defined in Chapter 4) which arise from post-modified noun phrases.

The DeReferencer has extensive knowledge of *sorts* (*qv*, Chapters 4 and 6), which represent collections of properties which entities in the discourse world may be said to exhibit (*eg* ball, red). It is able to make simple inferences about them and the relations between them. There are 'slots' in the existing implementation to allow the insertion of procedures to deal with (for example) inference from the preceding discourse (to allow this kind of 'practical' reasoning: "X has been put on a table" ⇒ "X is on the table"). As I explained before, this is not quite theoretically pure, because the representation language represents something like the surface form of the discourse, and not its semantics; strictly, this kind of inference should be made over some translation of the surface form into some suitable reasoning system. Anyway, this kind of reasoning is outside the scope of George's theory of reference, and therefore outside the scope of the work presented here.

The DeReferencer links any references found in the input discourse with a set of proposed possible referents (*ie* entity tokens), and in the event that the system fails to find a referent for a given reference, it creates one which is suitable (the criteria determining this suitability will be discussed in Chapter 6). This means that the dereferencing process is more likely to be unsuccessful because there are too many rather than too few possible referents for a reference.

Finally, there are a number of very simple rules which the DeReferencer applies to narrow down its candidate sets of entity tokens on a purely mechanical basis. For example, different referring expressions appearing within the same clause cannot be associated with the same referent; this can be expressed in simple syntactic terms in GRL.

On these bases, then, the DeReferencer makes judgements about what referring expressions may refer to what objects in the discourse world. Referential ambiguity is maintained, in accordance with George's fundamental hypothesis about maintainance of ambiguity in general – George at all times supposes that the 'speaker' (the user) is trying

to make sense to the 'hearer' (George) and that, in the event that a sentence is lexically or referentially ambiguous, the speaker realises as much and will attempt to resolve the ambiguity subsequently.

3.4. Representing Ambiguity

As was stated before, George deals with what is conventionally called lexical ambiguity in two different ways, and with referential ambiguity in a third.

Lexical ambiguity (by which I mean the existence in the lexicon of more than one translation of a word) is dealt with by the creation of new George processes – one for each possible interpretation – when it is impossible (or for some reason undesirable) to use a single ambiguous representation.

However, the George Parser also allows some of what might conventionally be viewed as lexical ambiguity to be maintained into the translation of the discourse. This is achieved by means of including deliberately ambiguous representations in the lexicon, coupled with a set of transformations applicable to syntax and semantics of translations of initial substrings of sentences. The transformations are guided by the syntax and sometimes the semantics of the partial sentence being transformed. They may be viewed as *coercions* – not unlike those defined in the Algol68 programming language; more on this in Chapter 5. The coercions allow ambiguous representations to be rewritten more specifically when (if) it becomes possible to do so.

Within any given translation, referential ambiguity (by which I mean ambiguity arising from inability of the hearer of a discourse to find a mapping for the referring expressions in it, in such a way that no referring expression is mapped to a wrong number of referents) is dealt with by allowing the existence of bindings between references and referents (using the terms informally) in any given translation which are one-to-many, as well as the more predictable many-to-one. This is equivalent to Charniak's and Mellish's candidate sets. It is not meaningful for a reference in one basic George system to be bound to a referent in another.

4. Summary

In this chapter, I have outlined three basic hypotheses relevant to the George system, and given a brief overview of the George system. The main points covered were as follows.

1. It is unnecessary, at least at a practical level, to represent "real" existence of entities in a discourse, as long as the participants are able not to disagree upon the properties

of those entities. This means that it is never necessary to refer to the real world; a model will always do.

2. There should be a separation between representation of discourse, discourse world and real world.

3. An interesting possibility, which may become a necessity in set reference analysis, is to make computational language systems able to reason about ambiguous representations. In general, this leads to a requirement of adaptability.

4. The George system is a strictly incremental parser and reference analyser, which attempts to represent ambiguity implicitly where possible, and deals with different readings in parallel otherwise.

5. Each basic George system theoretically consists of the following sections: Parser, DeReferencer, Discourse Memory, Entity List and Bindings. The last three constitute the representation of the discourse so far, which may be translated into First Order Predicate Calculus by a straightforward syntactic algorithm. In each basic George system, both Parser and DeReferencer must succeed, otherwise the process dies.

6. The George Representation Language (GRL) is used to represent the surface form of the input discourse adaptably.

7. Parsing is performed word by word; a self-contained (partial) representation of the input is always produced.

8. Mechanisms exist for the representation of sets, their subsets, and the relations of membership and cardinality between the two. Adaptability is fundamental to these.

5. Afterword

It is now necessary to flesh out the introduction which has been given in this chapter. The next four chapters contain details and formal specification of the three main parts of the system introduced in this chapter. These parts, and the theory they realise, will be summarised and related to comparable work in Chapter 8.

Chapter 4
The George Representation Language

Abstract

The George Representation Language is built on a basis structurally similar to the First Order Predicate Calculus. Its function within the George System is to provide a means of expression for the discourse being analysed, in order that the semantics of the discourse may be stored and related with the information about reference which George computes.

The Language differs from First Order Predicate Calculus in that it has explicit operators for certain linguistic concepts (such as definiteness); its usage differs from that of First Order Predicate Calculus in comparable Natural Language Understanding systems in that it is used to represent something close to the surface structure of the discourse, rather than its (subsequently calculated) semantics and, in particular, the reference therein.

The George Representation Language is presented, first informally as an introduction with motivation, then with full formality.

First, the basic concepts involved in the language are laid out in the context of representing reference. This leads into a series of examples showing how the language can be used to describe utterances in a form closer to the linguistic surface form than is conventional.

The main part of the chapter deals with the formal definition of the language.

The language's truth-conditional semantics is shown, through a conversion algorithm to FOPC.

(The manipulation of references – in particular of *indexed* references – and the rules and operations associated with it, are covered in Chapters 6 and 7.)

1. Introduction

The George Representation Language (GRL), the representation medium used by the George Parsing and Dereferencing System, is a quasi-logical first-order symbolic representation. It has been designed to be as much like First Order Predicate Calculus (FOPC) as possible, diverging only where a specific notation is deemed useful to the central aim of this research. Therefore, the parts of GRL outside FOPC are associated with expressing the form of referring expressions in English language.

This chapter presents first an informal introduction to the syntax of GRL and the intuitions behind it. This concludes with some examples of English sentences and their translations. The rest of the chapter gives the formal definition of the language, including a translation procedure for converting a complete George translation into an FOPC semantic representation.

2. Representing Surface Form

It is important to understand that GRL's function in George differs quite markedly from that of FOPC (or other logics) in many systems concerned with the same problems, in that it does not claim to represent the pure semantics of the input discourse, but something much closer to its surface structure (deliberately implicitly including some forms of ambiguity). For example, GRL permits explicit representations for different voices of verb, which might more conventionally be covered by inversion of arguments; and it includes explicit operators for such concepts as definiteness.

The reasoning behind this approach is fundamental to the philosophy of the George system; it is derived from the "surface form" hypothesis proposed in Chapter 3. In general, GRL is capable of representing explicitly all of the syntactic, semantic, and/or pragmatic features required to do the kinds of operations performed by the George DeReferencer (see Chapter 6). These features are: definiteness; number; discourse-world sort; and some cases of subordination.

It is desirable to maintain such an explicit representation of the (near-)surface form of the discourse, rather than of its calculated (and possibly wrong) semantics, in order that re-evaluation of the system's decisions on reference may take place if necessary, with the minimum of recalculation. In other words, we wish never to be forced to discard information by opting for a particular semantic interpretation of a surface construct, because if we do so, we sacrifice adaptability. For example, consider this variation on a pair of sentences used in [Webber, 1979]:

"Bonzo, the great Dane, ate the whole VW.	18a
The great Dane is a large dog.	18b
However, Bonzo is unusually small."	18c

If we were to opt for the reading of 18 where "the great Dane" in 18b refers to the same individual as that in 18a, and represent it semantically in Webber's style, as shown in 19, we would be forced either to fail or to backtrack on presentation of a subsequent sentence like 18c.

$$S_1{:}\exists(\text{Bonzo:greatDane}).\text{ate}(\text{VW},\text{Bonzo})$$
$$S_2{:}\exists(\text{Bonzo:greatDane}).\text{Large}(\text{Bonzo}) \;\&\; \text{Dog}(\text{Bonzo})$$
$$\text{Bonzo: } \iota(\text{x:greatDane}).\text{ate}(\text{VW},\text{x}) \;\&\; \textbf{evoke}(S_1,\text{x}) \;\&\; \textbf{evoke}(S_1,\text{x}) \qquad 19$$

This need not be a serious problem if the third sentence appears soon. But we have no way of knowing when or if it will do so. Therefore, to be foolproof, we need to be able to backtrack over the entirety of an arbitrarily large discourse translation procedure – and it is not hard to imagine a situation where the truth about Bonzo appears as a "punch-line" to quite a long (shaggy-dog?) story.

Further, as stated in the "ambiguity" hypothesis of Chapter 3, we are working under the assumption that it is useful deliberately to maintain any ambiguity which appears in a (partial) sentence without making any decisions about the correct translation. The practical reasons for this in terms of resolving syntactic ambiguity will become clear in Chapter 5; and [Mellish, 1981] has shown that it is useful to extend the principle to referential ambiguity also. For now, though, the relevant point is that a very good way to do this is simply to represent the ambiguous input directly. The part of GRL devoted to the annotation of referring expressions has been designed in such a way that all the aspects of plural and quantified reference may be explicitly and separately denoted, giving us a firm grip on the different aspects of any reasoning we may need to follow about the referent sets of referring expressions.

3. Representing Sentence Surface Structure

A surface representation of this kind also gives us a useful handle on the hierarchical structure of sentences. This is comparable with the idea presented in [Webber, 1979] where subordinate clauses are built up into *compound types*, as shown in 20a and explained more fully in Chapter 8:

"A man I saw":	$\exists(x : \lambda(y{:}Man)[Saw\ y, I]).x$	20a
"A fat man":	$\exists(x : \lambda(y{:}Man)[Fat\ y]).x$	20b

However, a single level of abstraction in the representation (as opposed to the extra one introduced by the λ-expression in 20) would be preferable, because of the complexity of these higher order terms. Further, the same notation does not allow a distinction to be made between adjectives and adnominals: compare 20a and b. I will discuss this further in Chapter 8.

Chapter 6 will show that we must have this information regarding sentence structure if we are to perform some of the more complicated reference analysis operations which George is designed to cover.

4. FOPC vs GRL

Except as detailed in the foregoing sections, the basic representational concepts underlying GRL are closely related to those of FOPC. The remaining important differences and similarities between the existing version of GRL and FOPC are as follows. Possible extensions to this limited version of GRL will be discussed in Chapter 9.

GRL does not include the existential quantifier, $\exists$, because existentials are dealt with outside GRL, by means of the binding mechanism (detailed in Chapters 6 and 7). GRL does include the universal quantifier, $\forall$, but allows it to quantify only over the set of non-negative integers (or natural numbers), $\mathbb{N}$. The $\times$ and $\otimes$ (*weak* and *strong index application*) operators allow this quantification to be passed to other elements of expressions in GRL. This will be explained in Chapters 6 and 7.

GRL includes a number of operators specially for expressing the information found in noun phrases. The index application operators mentioned above are two of these, used to indicate the number or quantification of a plural reference. The others are ! (*definite*), ˜ (*sort application*), and ▸ (*context extension* – the addition of the information in a subordinate clause to the associated reference in its superordinate clause).

A further binary operator, *coref*, enables us explicitly to state that two references have the same referent, as in, for example, equatives (*eg* "The big man is the doctor.").

GRL includes two *abstraction* operators, the common λ (lambda) and # (hash), the latter being specific to the construction of translations of NPs. Both are defined purely in terms of textual substitution. It will become clear that these operators are in some sense "second class" in that they do not have a semantics in the language proper – they only appear in the output of intermediate steps towards producing expressions with a full semantics, and cannot appear in the finished form of the translations.

The scoping of the universal quantifier in GRL is rather different from that in FOPC, in that the scope of a universal is simply the entirety of the expression in which it appears; quantification is thought of as part of a reference, not part of an expression. It is useful to perform an operation called *quantifier expansion*, to bring quantifiers to the front of expressions. This makes no semantic difference, because of the very loose scoping of the quantifier, but the result is quite a lot clearer to the human reader, and considerably easier to manipulate using first-order term unification, which George uses extensively. A

corresponding operation is applicable to expressions containing the context extension operator, ▸, for the same reasons.

5. Basic Concepts

5.1. Representation of Referring Expressions

In GRL, *references* to items in the discourse being represented are connected by *predicates* which express relations between the items.

Most of the interest of this part of this work lies in the references. The purpose of a George reference is to represent the information in the surface form of a noun phrase – and neither the referent of that NP nor its intension. It will be seen that this is particularly useful when dealing with references which are initially ambiguous. There are various kinds of reference, the kind of a given reference depending on the various operators which are applied to it.

One important feature of George references, with respect to adaptability, is that they contain information regarding the "discourse-world sort" of the entity to which the reference refers. Suffice it to say for now that these sorts are collections of properties, represented in GRL by the English word (or one of its synonyms) representing those properties. Thus, "a red ball" has sorts "red" and "ball". Sorts are arranged in a partial lattice (see Section 5.5, "The Sort Hierarchy"), which will allow us easily to analyse reference by synonym and/or partial specification.

The basic representation of referring expressions is the *simple reference*, like this:

ref1 21

(in general, "refN" where $N \in \mathbb{N}$). This denotes a indefinite referring expression conveying no sort information, if such a thing exists.

The simplest useful kind of reference is the *simple sorted indefinite reference*. This essentially represents singular (or generic) indefinite noun phrases. For example, the phrase "A man" translates as:

ref1¯man 22

The "man" part is a sort, and is applied by the ¯ (sort application) operator, like a selection operator choosing a subset of a set of possible referents, to the reference symbol.

The next simplest reference is the *simple sorted definite reference.* It is similar to the simple sorted reference, but for the application of the post-fixed definite operator, !. Thus, "The man" translates as:

ref1¯man! 23

Note that, however many sorts we apply to a reference (see below), the definite operator is always postfixed to the whole – that is, the sort application operator can never apply to a definite operator. This makes intuitive sense – the analogue in English would be a determiner appearing in between an adjective and its noun.

The next kinds of reference are those which are *indexed.* An example of a *weak indexed sorted definite reference* is the translation of "The men" shown in 24. The intuition is that the quantification is represented by "∀ind1", the domain of the quantifier being (some subset of) ℕ; the application of that quantification to the reference is represented by the application of "ind1" to the references by the × (*weak index*) operator.

∀ind1.ref1¯man!×ind1 24

Weak indexed indefinite sorted references differ from this only in that they lack the definite operator, as one might expect.

Strong indexed references are identical except that they contain the strong index operator, ⊗, instead of the weak one. We will see later that it is possible for a reference to contain both weak and strong operators – therefore, it is sometimes useful to think of a reference as being weakly or strongly indexed with respect to a particular index.

Indexed references denote noun phrases which refer to sets; it is the indexing mechanism and the operations one can apply to it which will provide the answer to the central question of this research. For the moment, it is enough to say that we will view references to sets as sets of references, by allowing the index (denoted by the index symbol immediately to the right of the quantifier) to range up from 0 over ℕ, and requiring that each reference/index pair under × and ⊗ denotes a different reference. In some cases (viz. sets with explicit cardinalities – eg "Two men"), we will apply an upper bound to the index variable. Indexing is comparable with the subscription found in [Mellish, 1981].

Lastly, *context extended references* are like the translation[1] of "The man who owns the donkey":

[owns,ref2~donkey!,ref1~man!] ▸ ref1~man! 25

The intention here is to represent the information found in the subordinate clause as a complete expression in GRL while maintaining its status as explicitly subordinate. This means that we are free to use the information in the subordinate clause either to select subsets of earlier references, or as "new" information. This, along with some simple assumptions about what can refer to what (covered in Chapters 6 and 7), will give us a means to deal with the distinction between restrictive and non-restrictive relatives.

5.2. Surface Structure

As stated above, one of the advantages of GRL with respect to adaptability is that it explicitly represents some of the surface structure of the sentences in a discourse – specifically that to do with subordination in noun phrases, by means of the context extension operator. This is comparable with [Webber, 1979]'s use of compound types – and justifiable by the same reasoning. For example, in example 25 above, the relative clause is identified as subordinate by its position to the left of the context extension operator.

Representing some surface structure in this way will prove to be useful when we consider dereferencing noun phrases containing relative and prepositional post-modifiers.

5.3. Predicates

Predicates in GRL are comparable with those in FOPC. The predicate modifiers used in George are intended as a hook on which to hang further work in tense, aspect, the effect of voice and mood on reference. Uncontroversially, they cover: tense, aspect, modality, voice, and mood, in that order. Thus, for example, "will eat" translates as the predicate

eat(present, perfect, future, active, indicative); 26

1: The predicate symbol "owns" is an abbreviation - the predicate in GRL contains information regarding tense, mood, etc, which is not relevant here.

while "may have been eating" translates as

eat(past, progressive, modal,active, indicative); 27

and "were being eaten" (as in "if it were being eaten") as

eat(past,progressive, conditional, passive, subjunctive). 28

The values available to the modifiers are:

tense	{past, present}
aspect	{progressive,perfect}
modality	{present, future, conditional, modal}
voice	{active, passive}
mood	{indicative, subjunctive, imperative, interrogative}

From here on in this document, for the purposes of explaining this work, predicates will be represented in a simpler form, just borrowing the appropriate form from the English input. Since most of the examples considered in this work are in the present tense, predicate names will usually be of the form of the third person singular of the verb, eg.:

eats. 29

5.4. Abstraction Operators

Of the two abstraction operators available in GRL, only # is unconventional.

Lambda, λ, may be used to form *λ-abstractions* like this:

$$\lambda x.f(x) \qquad 30$$

where $f(x)$ is an expression containing x. Such expressions may be *λ-applied to* other expressions by the simple textual substitution of the expression to which it is being applied for x in $f(x)$. Because these expressions are used only for the association of fillers with slots during semantic composition, the abstracted variable always appears in the body of the expression, as in Church's original λ-calculus ([Church, 1940], [Church, 1941]), now called λI- or sometimes λK-calculus.

Invariably, in the George Parsing and Dereferencing System, this *λ-reduction* is followed by an evaluation step. This evaluation (fully described in Section 7.1.4), however, is best

thought of as triggered by, rather than part of, the λ-reduction procedure. #-reduction entails no such evaluation.

Also, λ-abstracted expressions may be *λ-composed* in the following way

$$\lambda x.f'(x) \; + \; \lambda y.f''(y) \; \text{–composition→} \; \lambda y.f'(f''(y)) \qquad 31$$

where + means "is combined with" and –operation→ means "under operation to give". (Note that this operation is defined here for the purposes of explanation only. It does not arise in the George system.)

#-abstracted expressions are rather more complicated. First, a *#-abstraction* may be *#-applied* only to an element of the George sort hierarchy. Effectively, this means that it may only apply to translations of nouns and adjectives. Thus, a #-abstraction represents an isolated determiner or a headless noun phrase.

For example, the #-abstraction associated with the indefinite article, "A" is

$$\text{ref1}\#\text{ref1} \qquad 32$$

Note that this is not quite the same syntax as for the λ-abstraction: the bound variable (or, more correctly, *bound reference symbol*) precedes the abstraction operator. Analogously with λ-abstractions, the right hand side of the expression is referred to as the *body*. This example is particularly simple: in general, the right hand side will be a more complex reference containing the bound reference symbol.

#-reduction is similar to λ-reduction in that it operates through *#-application* to its argument, except for a special *#-composition* of adjectives.

For example, if we apply the translation of "A" to the translation of "man", we get the following behaviour (see Chapter 5 for an explanation of the actual application mechanism):

$$\text{ref1}\#\text{ref1} \; + \; \text{man} \; \text{–application→} \; \text{ref1}\,\bar{}\,\text{man} \qquad 33$$

On the other hand, with an adjective, for example, "male" (which has syntactic category n/n) we get

$$\text{ref1}\#\text{ref1} \; + \; \text{male} \; \text{–composition→} \; \text{ref1}\#\text{ref1}\,\bar{}\,\text{male}. \qquad 34$$

Finally, the composition of a λ-abstraction with a #-abstraction (or *vice versa*) is not allowed. The insertion of partial noun phrases into verb argument positions, which one might have expected composition to do, is performed by the protraction operation, defined in Chapter 5.

5.5. Sort Hierarchy

GRL supports a lattice of *sorts*, corresponding with the set of English nouns and adjectives. A sort is written in GRL as the appropriate noun or adjective. For the purposes of this work, the issues of representation of relative and intensional adjectives, though important in themselves, have been set aside.

Sorts are also represented by sets of property/value pairs, which correspond with the sort symbols but do not figure in GRL proper – the sets are used by the Parser and the DeReferencer for reasoning about semantic well-formation and reference respectively. This representation is more expressive than those used in some comparable systems, in that the connectives between the members of the pairs, rather than being the usual implicit "has value" are explicit and may be negated, thus admitting failure of consistency in a way equivalent to the notion of typed disunification (in the same way that consistency may be viewed as equivalent to typed unification). In particular, the value position may contain an uninstantiated variable, allowing the unform representation of specifically undefined values. This is also possible in, *eg*, KL-ONE of [Brachman & Smolze, 1985]

The basic connectives between properties and their values are *is* and *isnt*; these are binary operators, infixed between their operands, the property being on the left. For syntactic sugar, and to avoid the need for explicit representation of truth values, the unary forms of *is* and *isnt*, and their alternative forms, *has*, and *hasnt*, can be prefixed. Thus, in order to represent "a black bird", we need (among other properties and values) "colour is black" and "has wings" (*ie* "wings is true"). Similarly, to represent "a harmless animal" we might use "isnt dangerous " (*ie* "dangerous isnt true" or "dangerous is false").

Returning to the previous example, "The red ball", we could view the sorts involved as

{ colour is red }

and

{ shape is ellipsoidal }

and compose them by set union:

{ colour is red, shape is ellipsoidal }

Uninstantiated values in these sort definitions are written as __ (from the anonymous variable, in Prolog). For (a rather strained) example, we could use __ to represent the sort "colourless" as meaning "colour isn't anything", like this:

{ colour isnt __ }

These sets of property value pairs are called *defining sets*. They specify three relations between sorts: *subsumption*, *consistency*, and *contradiction*, as follows (full definitions are given in Section 7.2.3).

X subsumes Y (where X and Y are sorts – eg human, man) if and only if every pair in the defining set of X appears in the defining set of Y. This is equivalent to "is a supersort of" or "is a generalisation of" – though, thinking extensionally, the word "subsumes" is more usefully descriptive.

X is consistent with Y if and only if no pair in the defining set of X specifies a value conflicting with a pair in the defining set of Y.

X contradicts Y if and only if X is not consistent with Y.

5.6. Sort Application

The association of sort information with references in GRL is perhaps one of the more esoteric aspects of the language.

When a sort symbol is *applied* to a reference, as in examples 33 and 34 above, the *sort application* operator, ˜, is used to denote the connection between the two. It is important to understand that the resulting combination of symbols is then itself viewed as a reference; the reference is *constrained* by the application (*cf* [Mellish, 1985]). Therefore, if we apply another sort symbol to that reference (as will often happen in all but the most trivial NPs), we get another reference which is more strictly sorted. Application of a sort symbol to a reference corresponds with a (possibly null) reduction in its candidate set of referents.

For example, consider the composition of "A" with (the adjective) "red":

$$\text{ref1\#ref1} \;+\; \text{red} \;\text{–composition}\rightarrow\; \text{ref1\#ref1}\,\tilde{}\,\text{red} \qquad 35$$

The body of the resulting #-abstraction is a reference, just like the body of the initial one, except that it only refers to things which may (NB may – not must – because of George's open world assumption) have the property of redness.

Now, we go on to apply the result of 35 to (the noun) "ball". We get this:

$$\text{ref1\#ref1}\,\tilde{}\,\text{red} \;+\; \text{ball} \;\text{–application}\rightarrow\; \text{ref1}\,\tilde{}\,\text{red}\,\tilde{}\,\text{ball} \qquad 36$$

Ultimately, it makes no difference which way round we write the two sorts. Note that this is contrary to Webber's argument about structural requirements of a representation – I will discuss this in Chapter 9. A new sort application appears within the scope of any other operator (*viz* $\otimes$, $\times$, ! or $\blacktriangleright$) already in the reference. Thus we can translate "The red ball" as follows.

$$\begin{aligned}\text{ref1\#ref1!} \;+\; \text{red} \;+\; \text{ball} \;&\text{–composition}\rightarrow\; \text{ref1\#ref1}\,\tilde{}\,\text{red!} \;+\; \text{ball} \\ &\text{–application}\rightarrow\; \text{ref1}\,\tilde{}\,\text{red}\,\tilde{}\,\text{ball!}\end{aligned} \qquad 37$$

5.7. Indexing and the Expression of Reference to Sets

A large part of GRL is devoted to the representation of "set references" – that is, reference to a number of entities in the discourse via a single noun phrase. GRL does not attempt to deal with such linguistic devices as collective nouns: the view is that, for example, a flange of baboons is a singular entity, and should be referred to as such, and that dealing with collective nouns is a further level of linguistic abstraction, as it were, which would only obscure the central theme of this research.

We need a general notation for applying a predicate to each member of a set, which will allow us explicitly to sub-divide the sets and make further predications on the resulting subsets. Consider, for example, another partial discourse adapted from [Webber, 1979]:

"Jim bought two t-shirts. 38a

The blue one was faulty." 38b

Here, since the set of entities is small, there is no real reason why we should not represent it something like this (ignoring tense and reading ⊕ as "exclusive or"):

$$\text{bought(t-shirt1, Jim)} \wedge \text{bought(t-shirt2, Jim)} \wedge$$
$$((\text{ blue(t-shirt1)} \wedge \text{faulty(t-shirt1)}) \oplus (\text{ blue(t-shirt2)} \wedge \text{faulty(t-shirt2)})) \qquad 39$$

However, if we want to represent

"Jim bought two thousand t-shirts. 40a

The blue ones were faulty." 40b

we get an explosion of conjuncts if we try to use the rather simplistic approach of 39. If the cardinality of the set is not defined at all, as in

"Jim bought some t-shirts. 41a

The blue ones were faulty." 41b

that approach is even less helpful, since we cannot even write down an exhaustive list of possibilities.

I therefore suggest that we need to introduce an explicit separation between the number information contained in the discourse and the semantic representation of the entity(ies) being represented.

We can make such a separation by showing each reference to each set of entities (ie each noun phrase) in the explicit representation of its surface form, while representing the sense of each as a prototypical entity having no number. The prototypical entity is then linked to the reference by a *binding*, which is calculated and updated if necessary by the George DeReferencer (qv, Chapter 6). Relations between set cardinalities are represented by *Bounding Constraints* (see Chapter 7).

The translations into GRL, then, of the above partial discourses are as follows[1].

38: $\forall ind1 < 2.[bought, ref2\text{~}t\text{-}shirt \times ind1, ref1\text{~}Jim!\ ]$ 42a

$\forall ind2.coref(\ ref4\text{~}faulty, ref3\text{~}blue! \times ind2\ ).$ 42b

1: NB: I do not claim that this is a "correct" way of representing proper names. It is merely a way which works well enough for the purposes of experiment.

40: ∀ind1 < 2000.[bought, ref2~t-shirt × ind1, ref1~Jim!] 43a

∀ind2.coref(ref4~faulty, ref3~blue! × ind2). 43b

41: ∀ind1.[bought, ref2~t-shirt × ind1, ref1~Jim!] 44a

∀ind2.coref(ref4~faulty, ref3~blue! × ind2). 44b

the last of which, for example, might be translated back into English as

"The entity called Jim bought an unspecified number of entities of sort t-shirt. 45a

An unspecified number of the blue entities were faulty entities." 45b

(Note that GRL is incapable of denoting the association between the entities of the first sentence and those of the second; this side of things is handled, in the context of the Discourse State, by the DeReferencer.)

The above examples show clearly how GRL mirrors the surface structure of the utterance. In particular, note that the translations of the second of each pair of sentences are all the same, as are the originals in English; contrast this with example 39 and its extensions to the harder problems in 40 and 41. I suggest, therefore, that this is perhaps a more correct translation of the pure sense of the sentence than, for example, the FOPC-style translation suggested in 39, because it does not include results of inference about references in the utterance.

5.8. Bindings and the Representation of Entities

The fine detail of the entities and bindings will be covered with the DeReferencer in Chapters 6 and 7. However, for clarity here, an example is appropriate.

First, two informal definitions. Outside GRL, George uses symbols called *entity tokens*, constructed from *entity token symbols* and sorts, to represent sets of entities in the discourse. Entity token symbols are of the form eN where N ∈ ℕ; sort symbols may be applied to entity token symbols using the same notation as for references. For example, an entity token representing the sense of "A donkey" would look like this:

e1~donkey

It is important to understand that the entity token does not directly represent the discourse entity itself, but some abstraction of its sense; the further level of *specification* indirects to the discourse world proper. Thus, the token represents an approximation of

the entity referred to in the discourse in the mind of the hearer. Recall also that an entity token may specify a discourse entity which has no corresponding entity in the real world (so we can discuss unicorns and such with impunity). It also means that an entity token can be viewed as denoting a generic or prototypical entity (*cf* prototypes in [Webber, 1979]). This ambiguity closely parallels that in English (which is desirable, according to George's design philosophy).

Entity tokens are explicitly associated with the references giving rise to and referring to them by structures called *bindings*. Bindings are ordered pairs, the first element being a set of co-referential references, as written in GRL, and the second being a non-empty set of entity tokens. Both sets in the pair are often singleton.

Consider again example 41. When we have translated 41a to give 44a, we may draw a diagram like figure 3a to represent the bindings and entity tokens. The arrows in figure 3 represent bindings between references and sets of entity tokens, and may be read as "refers to". The sentences and translations are reproduced here:

(	"Jim bought some t-shirts.	41a
	The blue ones were faulty."	41b)
(	∀ind1.[bought, ref2˜t-shirt×ind1, ref1˜Jim!]	44a
	∀ind2.coref(ref4˜faulty, ref3˜blue!×ind2).	44b)

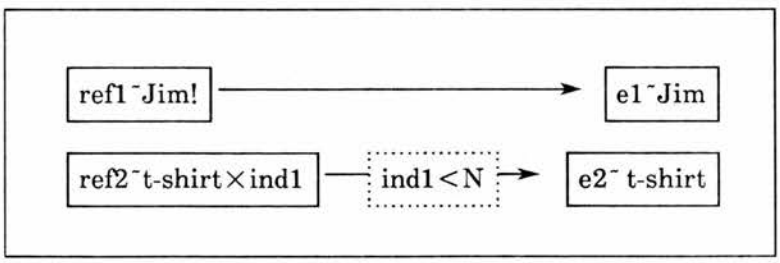


Figure 3a: Entities and Bindings for Example 41/44a

Since the ∀ quantifier is viewed as part of any reference with which its bound index is applied (by × or ⊗ – see Section 7.2), the information it carries must appear in the binding. In the figures 3, where the reference part of the bindings is shown in the boxes at the blunt end of the arrows, quantification is represented by the application of the × operator and an index. The information conveyed by the upper bound of the binding is then shown in the box attached directly to the arrow. The set of entity tokens at the other end of the binding is at the sharp end of the arrow.

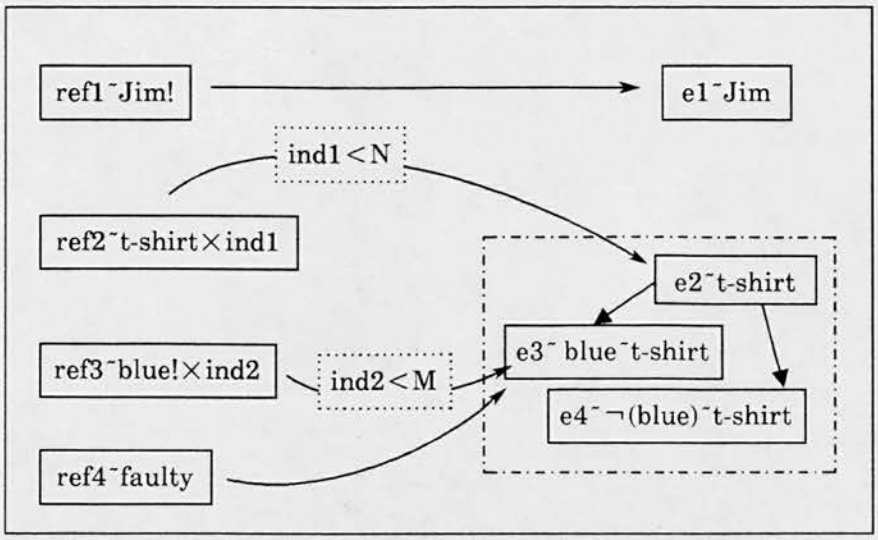


Figure 3b: *Entities and Bindings for Example 41/44b*

In the case of example 41/44, the upper bounds on the quantifications are unspecified. Therefore, in the diagrams, the upper bound is written is denoted by a variable, N or M, whose value is in ℕ. Such unknowns may be denoted explicitly in GRL by ⊥.

The attachment of the variables to the arrows may be thought of as shorthand for N or M bindings from N or M distinct but characteristically identical references to N or M distinct copies of the entity token (these notions will be formalised as *entity token partition* in Chapter 6, and *index expansion* and *index partition* in Chapter 7, where I will return to this example). In a case such as this where N and M are unspecified, it is (still) impossible to draw the complete expansion.

6. Some Examples of Expressions in GRL

This completes our informal introduction to GRL. Below are some examples of English sentences and their translations in GRL. It is important to remember that GRL is intended to represent the surface form of the utterance, and so does not, for example, allow substitutions of referents for references once reference has been determined. Remember that the syntactic differences between FOPC and GRL lie almost entirely with references (instead of logical variables in FOPC); this is deliberate design strategy, since the language is intended as a vehicle for research into reference. A Backus-Naur Form specification of the syntax of GRL is given in Appendix D.

"A man owns a donkey" [owns,ref2~donkey,ref1~man]

"The man beats the donkey" [beats,ref2~donkey!,ref1~man!]

"The donkey hates the man who beats him"

[beats,ref3~male!,ref2~man!] ▸ [hates,ref2~man!,ref1~donkey!]

"Jim is a man" coref(ref1~Jim!, ref2~man)

"Fred is a man" coref(ref1~Fred!, ref2~man)

"The men own a donkey" $\forall$ind1.[owns,ref2~donkey,ref3~man!$\times$ind1]

"Pedro is the donkey" coref(ref1~Pedro, ref2~donkey!)

"Every man owns a donkey" $\forall$ind1.[owns,ref2~donkey$\otimes$ind1,ref3~man!$\otimes$ind1]

"Every man who owns a donkey beats it"

$\forall$ind1.[owns,ref2~donkey$\otimes$ind1,ref3~man$\otimes$ind1]

▸ [beats,ref2~it!$\otimes$ind1,ref3~man$\otimes$ind1]

"Two men own a donkey" $\forall$ind1$<$2.[owns,ref2~donkey,ref3~man$\times$ind1]

"One man feeds him" $\forall$ind1$<$1.[feeds,ref3~male!,ref1~man!$\times$ind1]

7. The Definition of GRL

7.1. The Syntax of the George Representation Language

7.1.1. Basic Categories and Functions

The elements of GRL are divisible into a number of basic categories according to function.

Var	variable symbols (countably infinite)	{var0,var1,...}
Ref	reference symbols (countably infinite)	{ref0, ref1...}
Ind	index symbols (countably infinite)	{ind0, ind1...}
Sort	sort symbols (finite)	{chair, man...}
Pred	predicate symbols	{beat, own, hate,...}
Mod	predicate modifier symbols (small)	{tense, mood..}
ModVal	modifier value symbols (small)	{past,present,subjunctive,indicative..}
Dom	a mapping from Mod to 2^{ModVal}	
Arity	a mapping from Pred$\times\Pi_{m\in\text{Mod}}$Dom(m) to $\mathbb{N}$	

(Var, Ref and Ind are in explicit bijection with $\mathbb{N}$.)

7.1.2. Syncategorematic Operators and Quantifier

Referential Control Operators

˜	binary sort application (most binding)
!	unary (postfix) definite
×	binary weak index application
⊗	binary strong index application
coref	binary coreference association
▸	binary context extension (least binding)

Quantifier – Forall (∀)

∀ associates a unique element of Ind (an *index*, which is *bound* by the quantifier) with a (well formed) reference and an optional element of ℕ. The entire GRL expression is the *scope* of the quantifier, and the element of Ind in question may appear only within this scope. Because of the quantifier expansion operation, which is explained below, the scope of a particular ∀ is the entirety of the expression in which it appears.

The natural number is an upper bound on the range of the index, which may in certain circumstances be thought of as a variable ranging over ℕ. This is part of the solution to our central question of subset reference.

A quantified expression (fully expanded) is written:

$$\forall \text{indN} < \text{M.Expr}$$

where $N \in \mathbb{N}$, $M \in \mathbb{N} \cup \{\perp\}$ (indN being an element of Ind), and Expr is a well formed GRL expression. In the (common) case where indN has no explicit upper bound, the "$< \perp$", which would explicitly represent the fact, may be omitted.

Abstraction Operators

Lambda (λ)

Given a variable symbol and an expression in GRL, denotes a λ-abstraction (see below).

Hash (#)

Given a reference symbol and an expression in GRL, denotes a #-abstraction (see below). #-abstraction is like λ-abstraction, except that the abstracted entity is a sort, and not a value.

7.1.3. Syntactic Well-Formation in GRL

A sentence created from the above elements of the language is syntactically well-formed if and only if it may be synthesised from the categories described above in any of the following ways.

A *Simple Reference* is any element of Ref. For example: ref1

A *Sorted Reference* is any simple or sorted reference combined by the sort association operator, ˜, with an element of Sort. For example: ref1˜man
ref2˜dog˜brown

A *Definite Reference* is any simple or sorted reference to which is applied the definite operator, !. For example: ref1˜man!

An *Indexed Reference* is any simple, sorted or definite reference associated by one of the index operators, × and ⊗, with a member of Ind, and preceded by a quantifier, bounded or otherwise, binding that member of Ind. For example: ∀ind1.ref1˜man!×ind1
∀ind2<4.ref2˜donkey!⊗ind2

A *Predicate* is any element of Pred associated with a tuple of predicate modifier values chosen from ($\Pi_{f \in Mod}$ Dom(f)). For example: run(present,perfect,present,active,indicative)

This will often be abbreviated here to the English verb form which carries the same meaning – in this example, "runs".

A *Closed Predicate* is any Predicate and a number (Arity(P) where P ∈ Pred and the Predicate is formed from P) of well-formed references separated by commas and enclosed in square brackets. For example: [run(present,active,continuous,indicative),ref1˜man!]
[eat(past,passive,perfect,subjunctive),∀ind1.ref1˜carrot×ind1]

A *Context Extended Reference* is a well-formed closed predicate combined with a reference by the context extension operator, ▸. The closed predicate to the left is the *context*

extension. The reference always appears inside the closed predicate. For example:

[own(present,active,continuous,indicative),ref2~donkey,ref1~man] ▸ ref1~man

A *Context Extended Expression* is a combination of two well-formed closed predicates by the context extension operator, ▸. For example:

[own(present,active,continuous,indicative),ref2~donkey,ref1~man] ▸
[beat(ref1~manpresent,active,continuous,indicative),ref3~it!,ref1~man]

These expressions are produced by quantifier expansion of a closed predicate with a context extended references as one of its arguments. The context extension must always have a reference symbol in common with the body (the closed predicate to the right of the operator), because of the nature of the linguistic behaviour – subordination – which context extension represents.

A *Hash (#) abstraction* is a reference prefixed by the # operator and an element of Ref, which is called a *bound* reference symbol. For example: ref1#ref1~green!

A *λ-abstraction* is an expression prefixed by the lambda abstraction operator and a bound variable (element of Var) which would be well-formed if all occurrences of the bound variable were replaced with a well-formed expression. For example: λvar1.var1

A *Coreference* is the combination of two well-formed references by the coref operator. For example, coref(ref2~bear, ref1~animal!)

7.1.4. Operations

Reduction

Given an abstracted expression and an expression of the appropriate kind (see below), reduction enables the creation of a new expression. After reduction, resulting expressions are evaluated (see below).

Lambda (λ) Reduction

A λ-abstraction may be *applied* to any other expression. The result of the application is obtained by replacing all occurrences of the bound variable in the body of the λ-abstraction with the expression to which it is being applied. The lambda operator and its bound variable do not appear in the result. For example:

λvar1.[pred,var1] + ref1 –application→ [pred,ref1]

A λ-abstraction may be *composed* with any other abstracted expression. The result of the composition is obtained by replacing all occurrences of the bound variable in the body of the λ-abstraction with the body of the expression with which it is being composed. The abstraction operator(s) of the latter expression are then prefixed to the result; the lambda operator and bound variable from the applied expression do not appear in the result. Recall that composition is only defined here for the purposes of discussion – it does not appear in the George system. For example:

$$\lambda var1.[pred1,var1] \; + \; \lambda var2.[pred2,var2] \; \text{–composition}\rightarrow \; \lambda var2.[pred1,[pred2,var2]]$$

Hash (#) Reduction

A #-abstraction may be applied to any member, S, of Sort. The result of the application is obtained by replacing all occurrences of the bound reference, R, in the body of the #-abstraction with a sub-expression of the form R~S. The # (hash) operator and its bound reference do not appear in the result. For example:

$$ref1\#ref1 \; + \; man \; \text{–application}\rightarrow \; ref1\sim man$$

A #-abstraction may be composed with any member of Sort. The result of the composition is obtained by applying the #-abstraction, and then prefixing the hash operator and its bound reference to the result. For example:

$$ref1\#ref1 \; + \; red \; \text{–composition}\rightarrow \; ref1\#ref1\sim red.$$

Quantifier Expansion

It is possible to make statements of the form:

$$[pred,\forall ind1.ref1 \times ind1]$$

in GRL. The statement is synonymous with:

$$\forall ind1.[pred,ref1 \times ind1]$$

Because GRL has no existential quantifier, ∃, and because the quantifier is viewed as modifying the whole sentence, we can make this purely textual transformation with impunity, even over multiple quantifiers.

Quantifier expansion is the operation which enables the scope of an arbitrary number of quantifiers to be expanded to cover the whole of the expression (except for abstraction

operators and their bound variable or reference) in which they occur, to form an exactly equivalent expression.

Quantifier expansion may also apply to expressions including the reference extension operator, ▸. The operator and its preceding expression may be moved so that they have the same scope as the innermost quantifier or abstraction operator in whose scope they appear.

Normally, quantifier expansion is applied by the parser immediately after each evaluation, to transform the resulting expression into one more easily readable and more easily manipulable under first order term unification.

Evaluation

Evaluation is the operation which enables the completion of reductions made possible by other reductions. For example, in the application shown below, a λ-abstraction is moved "inside" another; the result must then be evaluated to enable the "inside" expression to apply to its (new) argument. Note that this example shows an application and not a composition, as one might at first think. The decision on whether to apply or compose is taken by the syntactic mechanism of the parser, described in Chapter 5.

λ(var1, [var1,ref1]) + λ(var2, [pred,var2]) → [λ(var2, [pred,var2]),ref1] → [pred,ref1]

Evaluation is recursive; the (unsurprising) results of applying it are as follows:

Op Exp –evaluation→ Op NewExp
Exp Op –evaluation→ NewExp Op

where Op is a unary operator and Exp and NewExp are well formed expressions in GRL. NewExp is formed by evaluating Exp.

Exp1 Op Exp2 –evaluation→ NewExp1 Op NewExp2

where Op is a binary operator and ExpI and NewExpI are well formed expressions in GRL. NewExpI is formed by evaluating ExpI.

[AbsExp,Arg] –evaluation→ NewExp

where AbsExp is an abstracted expression, and Arg and NewExp are well-formed expressions in the language, and NewExp is formed by applying AbsExp to Arg.

$\forall$Index.Exp $\xrightarrow{\text{evaluation}}$ $\forall$Index.NewExp

where Index $\in$ Ind, Exp and NewExp are well formed expressions in GRL, and NewExp is formed by evaluating Exp.

Evaluation for all other expressions is the identity function.

7.2. Semantics and the Discourse World

7.2.1 Introduction

In George, the representation of the discourse is split into two parts. First, there is the Discourse World, which defines the properties of objects (via their sorts) and in which Discourse Entities exist (though they are accounted for neither here nor in the implementation). The meaning of predicates defining actions and states is defined here. Second, there is the Discourse State, in which predications about the discourse world is made. The Discourse State therefore is defined in terms of and within the Discourse World, and is meaningless outside it. The Discourse State includes the set of Entity Tokens which specify the Discourse Entities. The set is partially ordered in correspondence with the $\supseteq$ relation. The Discourse State includes translations of the input discourse, bindings from the references in it to the entity tokens in the Discourse World, and Bounding Constraints which, syntactically, are arithmetic equations.

7.2.2. The Discourse World

A *Discourse World* consists of

- a finite set, Prop, of property symbols {colour, species,...}
- a countably infinite set, PropVal, of property value symbols {green, true, 3,..}
- a small set, PropConn, of property/value connectives {is, isnt}
- a set, WDef $\subseteq 2^{\text{Prop}\times\text{PropConn}\times\text{PropVal}}$ {⟨colour, is, red⟩, ... }
- a function[1], SortDef, between Sort (*ie* the sort symbols in GRL) and WDef
- a set, DEnt, of discourse entities

1: If this function is a bijection, certain efficiency measures can be taken in the consistency checking algorithms; this option is built in to the existing system.

Given a Discourse World, a *Discourse State* consists of

- an ordered set, Sense, of syntactically well-formed GRL expressions
- a function, Bind, from 2^{SRef} to 2^{Ent}

  where SRef is the set of references which appear in expressions in Sense
- a relation, HasSort, between Ent and Sort
- a function, IndexRange, from Ind to $\mathbb{N} \cup \{\perp\}$
- a set, Bound $\subseteq$ IExpr $\times$ Ind

  where IExpr = Ind $\cup$ (IExpr $\times \{+,-\} \times$ Ind)
- a set, Ent, of Entity Token Symbols
- a partial order, $\gg$, on Ent

Utterances are regarded as mapping from discourse states to discourse states; a discourse is seen as starting with some initial discourse state (possibly empty) and updating that state with new elements of Sense and possibly new information defining HasSort, IndexRange and/or bind. If Sense contains no contradictory groups of expressions, directly or by inference, it is consistent; the Sense of a discourse state may correctly be inconsistent. IndexRange specifies upper bounds on indices; the set Bound specifies arithmetic relations between members of Ind.

7.2.3. Relations on Sorts

$\forall s_1.\forall s_2.\ s_1 \in \text{Sort} \wedge s_2 \in \text{Sort} \Rightarrow (\ s_1$ **subsumes** $s_2 \Leftrightarrow \text{SortDef}(s_2) \supseteq \text{SortDef}(s_1))$

$\forall s_1.\forall s_2.s_1 \in \text{Sort} \wedge s_2 \in \text{Sort} \Rightarrow (\ s_1$ **strictly subsumes** $s_2 \Leftrightarrow \text{SortDef}(s_2) \supset \text{SortDef}(s_1))$

$\forall s_1.\forall s_2.s_1 \in \text{Sort} \wedge s_2 \in \text{Sort} \Rightarrow$

$(\ s_1$ **is consistent with** $s_2 \Leftrightarrow$

$\forall p.\forall v.\forall q.\forall u.(\langle p,is,v\rangle \in \text{SortDef}(s_1) \wedge \langle q,is,u\rangle \in \text{SortDef}(s_2) \wedge (p=q) \Rightarrow (u=v))$

$\wedge\ \forall p.\forall v.\forall q.\forall u.(\langle p,is,v\rangle \in \text{SortDef}(s_1) \wedge \langle q,isnt,u\rangle \in \text{SortDef}(s_2) \wedge (p=q) \Rightarrow (u \neq v))$

$\wedge\ \forall p.\forall v.\forall q.\forall u.(\langle p,isnt,v\rangle \in \text{SortDef}(s_1) \wedge \langle q,is,u\rangle \in \text{SortDef}(s_2) \wedge (p=q) \Rightarrow (u \neq v))$

$\forall s_1.\forall s_2.s_1 \in \text{Sort} \wedge s_2 \in \text{Sort} \Rightarrow (\ s_1$ **contradicts** $s_2 \Leftrightarrow \neg\ (\ s_1$ **is consistent with** $s_2))$

8: Translation Algorithm:
GRL to First Order Predicate Calculus

8.1. The Target Language and the Translation Procedure

In order to make George worthwhile in a wider context, I must supply a semantics for its output in terms of a standard (or at least well-understood) formalism. The formalism chosen here is First Order Predicate Calculus. The choice of such a language highlights certain issues already discussed in this document. In particular, GRL has operators which allow the explicit statement of information about reference and other extra-logical issues (*eg* definiteness and subordination) and there are Bindings linking the representation to a model of the discourse world. The translation algorithm, therefore, discards this information, and represents only the truth-conditional aspects of the input discourse in its output. It is fundamental to the operation of the George system that such a loss of information should not be irrevocable, or that it should occur only when the discourse has definitely finished, because otherwise the adaptability of the representation (and therefore George's ability to resolve any existing ambiguity) would be compromised. Thus, the FOPC output, which can be generated from any set of complete GRL expressions, entity tokens and bindings, should be used for reasoning about the truth-conditional aspects of the discourse in parallel with George's operation and without sacrificing the extra-logical information encoded in the system. It is also possible to imagine a system where GRL predicates have analogues at the Entity Token level, in which case discourse world inference might proceed at that level – this might lead to an analysis of sentential anaphora; events could be represented by entities, as in situation semantics ([Barwise & Perry, 1983]).

The target language for this transformation is FOPC with the inclusion as predicates of set operations $\in$, $\supseteq$, $\|$ (modulus), and $=$ (equality). A naming isomorphism, `, is used, which, given a natural number or a logical object in GRL or FOPC, produces an isomorphic string naming that object; the function is reversible. I also use the usual function, gensym, which generates unique object-level logical symbols (here, exclusively FOPC variables).

The translation procedure is specified here in three main parts. First, I translate the discourse world (whose translation will be the same for any discourse state within that

world). Second, the entities and sets referred to in the discourse must be introduced. Finally, the expression defining the discourse state must be introduced.

The neatest way to defined this transformation is in terms of meta-logical functions, whose behaviour is determined by logical expressions involving the sets defining the discourse world and state, and whose output is a string naming the ultimate output, rather than the output itself. Because of the reversible nature of the naming isomorphism, `, this string can then be "de-named" to produce the FOPC translation; this is the technique used extensively in meta-level reasoning in the logic programming community (see *eg* [Lloyd, 1987], [Kowalski, 1979]). The string-based nature of the definition requires the use of a concatenation operator, +. The definitions will also use the functions Pick and First. Pick chooses an arbitrary member of a set; First chooses the first item of an ordered set or one of the first items of a partially ordered set. We also need one new logical object and two meta-level functions not already in the definition of GRL. These are:

Predicates: a set of FOPC objects in bijection with $\text{Pred} \times \Pi_{m \in \text{Mod}} \text{Dom}(m)$

PredOf: a function over GRL closed predicates which returns the member of Predicates corresponding with the modified predicate symbol in the argument expression.

RefOf: a function over GRL closed predicates which returns a set containing the references (including quantifiers) in the argument expression, ordered in the same sequence as in the expression.

Because the translation specified here is a recursive functional algorithm, rather than a relation, it is quite hard to read. Therefore, an informal specification and an example of the output of each main function is given. The translation algorithm has been implemented, and output examples are given below and in Appendix B.

8.2. Examples of George Output

Figures 4a and b show the actual George output for two different discourses, given in examples 46a and b. The FOPC translation is shown in the figures between two broken lines. The output of the program has been massaged slightly to make it more readable: symbols like "forall" and "=>" have been replaced with their more conventional equivalents, and the layout has been changed (by the addition of indentation) to reflect quantifier scope.

```
** George parser **

State 1:

** s:∀ind1.∀ind2<2.[own(pres,pres,perf,act,indic),
                        ref2~donkey×ind2,ref1~man×ind1]
** s:[own(pres,pres,perf,act,indic),ref5~carrot,ref4~woman]
** s:∀ind4.∀ind5.[give(pres,pres,perf,act,indic),
                        ref9~donkey!×ind5,
                        ref10~carrot!,
                        ref7~person!×ind4]

ref1 is bound to e1~man
ref2 is bound to e2~donkey
ref4 is bound to e4~woman
ref5 is bound to e5~carrot
ref7 is bound to e4~woman
       and e1~man
ref9 is bound to e2~donkey
ref10 is bound to e5~carrot

------------------------------------------------------------
∀x.(carrot(x) ⇒ colour(x,orange)) ∧
∀x.(carrot(x) ⇒ nature(x,vegetable)) ∧
∀x.(donkey(x) ⇒ nature(x,animal)) ∧
∀x.(man(x) ⇒ gender(x,masculine)) ∧ ∀x.(man(x) ⇒ type(x,human)) ∧
∀x.(man(x) ⇒ nature(x,animal)) ∧
∀x.(woman(x) ⇒ gender(x,feminine)) ∧
∀x.(woman(x) ⇒ type(x,human)) ∧ ∀x.(woman(x) ⇒ nature(x,animal)) ∧
   ∃v1.(carrot(v1) ∧ ∃v2.(woman(v2) ∧
   ∃V1.(|V1|=2 ∧ (∀v3.v3∈V1 ⇒ donkey(v3)) ∧
   ∃V2.((∀v4.v4∈V2 ⇒ man(v4)) ∧
      (∀v5.(v5∈V1 ⇒ ∀v6.(v6∈V2 ⇒ own(v5,v6)))) ∧ (own(v1,v2)) ∧
      (∀v7.(v7∈V1 ⇒ give(v7,v1,v2)) ∧
         ∀v7.(v7∈V1 ⇒ ∀v8.(v8∈V2 ⇒ give(v7,v1,v8))))
   ))))
------------------------------------------------------------

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

>> ^D !! End of input Stream !!

   ** Terminating parse **
```

Figure 4a: *George output with FOPC translation for example 46a*

The examples demonstrate various points about the translation. 46a shows quantification applying sort predicates to members of sets of known and unknown cardinality; it shows singular reference; and it shows conjunction arising from reference to members of a set composed of an individual and another set split into individuals by quantification.

```
** George parser **

State 1:

** s:[own(pres,pres,perf,act,indic),ref2~donkey,ref1~man]
** s:[own(pres,pres,perf,act,indic),ref5~carrot,ref4~woman]
** s:[give(pres,pres,perf,act,indic),
                  ref9~donkey!,ref10~carrot!,ref7~person!]

ref1 is bound to e1~man
ref2 is bound to e2~donkey
ref4 is bound to e4~woman
ref5 is bound to e5~carrot
ref7 is bound to e4~woman
          or to e1~man
ref9 is bound to e2~donkey
ref10 is bound to e5~carrot

------------------------------------------------------------
∀x.(carrot(x) ⇒ colour(x,orange)) ∧
∀x.(carrot(x) ⇒ nature(x,vegetable)) ∧
∀x.(donkey(x) ⇒ nature(x,animal)) ∧
∀x.(man(x) ⇒ gender(x,masculine)) ∧ ∀x.(man(x) ⇒ type(x,human)) ∧
∀x.(man(x) ⇒ nature(x,animal)) ∧
∀x.(woman(x) ⇒ gender(x,feminine)) ∧
∀x.(woman(x) ⇒ type(x,human)) ∧ ∀x.(woman(x) ⇒ nature(x,animal)) ∧
   ∃v1.(carrot(v1) ∧ ∃v2.(woman(v2) ∧
   ∃v3.(donkey(v3) ∧ ∃v4.(man(v4) ∧
      (own(v3,v4)) ∧ (own(v1,v2)) ∧
      (give(v3,v1,v2) ∨ give(v3,v1,v4))
   ))))
------------------------------------------------------------

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

>> ^D !! End of input Stream !!

   ** Terminating parse **
```

Figure 4b: *George output with FOPC translation for example 46b*

"Some men own two donkeys. A woman owns a carrot.
The people give the carrot to the donkeys." 46a

46b is deliberately ambiguous. It shows the disjunction introduced by unresolved singular references.

"A man owns a donkey. A woman owns a carrot.
The person gives the carrot to the donkey." 46b

8.3. World Formation

To translate the Discourse World, we translate those (parts) of its defining sets containing information used in the final representation: Prop, PropVal, Sort and WDef. We will also need to use the function SortDef. The translation function TWorld calls the other translation functions, and is called initially with the expression

TWorld(Sort, Ent, Sense)

the last two arguments being passed to deeper function calls, which are defined below.

Specification:

For each sort symbol, s, in the discourse world (*ie* in the domain of SortDef),
there is a predicate, P, in FOPC, with the same name such that,
if P is true of a FOPC object, x, then
for each ⟨Property,is,Value⟩ triple in SortDef(s),
the FOPC relation Property holds of the pair ⟨x,Value⟩, and
for each ⟨Property,isnt,Value⟩ triple in SortDef(s),
the FOPC relation Property does not hold of the pair ⟨x,Value⟩.

Definition:

If S = ∅ then TWorld(S, SE, Se) = TDomain(SE, Se, ∅).
Otherwise, let s be Pick(S). Then
TWorld(S, SE, Se) =
TWorld'(s, { ⟨p,c,v⟩ : ⟨p,c,v⟩∈SortDef(s) }) + TWorld(S \ { s }, SE, Se)

If SD = ∅ then TWorld'(s, SD) = "".
Otherwise, let ⟨p,c,v⟩ be Pick(SD). Then
If `c = "is" then
TWorld'(s, SD) = "∀x.(" + `s + "(x)" + "⇒" + `p + "(x, " + `v + ")) ∧"
+ TWorld'(s, SD \ { ⟨p,c,v⟩ }).
If `c = "isnt" then

TWorld'(s, SD) = "∀x.(" + `s + "(x)" + "⇒ ¬" + `p + "(x, " + `v + ")) ∧"
+ TWorld'(s, SD \ {⟨p,c,v⟩}).

Example:

A world in which carrots are distinguished from other entities by being orange and edible.

Let Sort = { carrot } and SortDef(carrot) = {⟨colour,is,orange⟩, ⟨edible,is,true⟩}. Then,

TWorld(Sort, Ent, Sense) =
"∀x.(carrot(x) ⇒ colour(x, orange)) ∧ ∀x.(carrot(x) ⇒ edible(x, true)) ∧"
+ TDomain (Ent, Sense, ∅)

8.4. Domain Formation

Domain formation is the translation of entity tokens existing in the discourse to variables in FOPC. It is defined in terms of the discourse state, using the sets Ent, Ind and Bind, the function IndexRange, the relation HasSort, and the partial ordering, ≫. The definition is in the form of a meta-logical function defined inductively over the partially ordered set, Ent. The meta-logical functions TSort, TPartition and TSense are defined below. The argument C is an accumulator, allowing us to keep track of which GRL entity token is associated with which FOPC variable. The FOPC representation of the existing GRL discourse state is obtained by the evaluation of TDomain(Ent, Sense, ∅), and is then placed in context of discourse world information by TWorld, above.

Specification:

For each entity token, e, in Ent,
there exists an existentially quantified FOPC variable, V, such that
if e is only bound to singular references,
V is a variable over individuals; for each element, s, of HasSort(e), s(V) holds.
if e is bound to plural references,
V is a variable over sets;
for each element v of V, for each element s of HasSort(e), s(v) holds;
if r is in R where e∈Bind(R) and r is indexed by I and IndexRange(I) ≠ ⊥, then
|V| = IndexRange(I) holds

Definition:

If SE = ∅, TDomain(SE, S, C) = TSense(S, C).

Otherwise, let e be First(SE). Then

If ∃⟨R,E⟩.⟨R,E⟩∈Bind ∧ e∈E ∧ ∀r.(r∈R ⇒ r is simple) or

∃⟨R,E⟩.⟨R,E⟩∈Bind ∧ e∈E ∧ ∃r.∃i.(r×i∈R ∧ IndexRange(i)=1) then

Let v = gensym;

TDomain(SE, S, C) =

"∃" + v + ".(" + TSort({ s : HasSort(e, s) }, v)

+ TPartition(e, v, C) + TDomain(SE \ { e }, S, C ∪ { ⟨ e, v ⟩ }) + ")".

If ∃⟨R,E⟩.⟨R,E⟩∈Bind ∧ e∈E ∧ ∃r.∃i.(r×i∈R ∧ IndexRange(i) = ⊥) then

Let V = gensym; let v = gensym;

TDomain(SE, S, C) =

"∃" + V + ".((∀" + v + "." + v + "∈" + V + " ⇒

+ TSort({ s : HasSort(e, s) }, v) + ")"

+ TPartition(e, V, C) + TDomain(SE \ { e }, S, C ∪ { ⟨ e, V ⟩ }) + ") ".

If ∃⟨R,E⟩.⟨R,E⟩∈Bind ∧ e∈E ∧ ∃r.∃i.∃N.(r×i∈R ∧ N∈ℕ ∧ IndexRange(i)=N) then

Let V = gensym; let v = gensym;

TDomain(SE, S, C) =

"∃" + V + ".(|" + V + "|= " + `N + "∧ (∀" + v + "." + v + "∈" + V + " ⇒ "

+ TSort({ s : HasSort(e, s) }, v) + ")"

+ TPartition(e, V, C) + TDomain(SE \ { e }, S, C ∪ { ⟨ e, V ⟩ })+ ") ".

Example:

A discourse consisting of "A man owns two donkeys. The man beats one donkey.". (Note that the calls to TSort and TPartition (see below) are also evaluated here for clarity.)

Let SEnt = { e1, e2, e3, e4 }, HasSort(e1) = { man }, HasSort(e2) = { donkey },
HasSort(e3) = { donkey }, HasSort(e4) = { donkey }, e2 ≫ e3, e2 ≫ e4, and
Bind = { ⟨ { ref1 ˜man }, { e1 } ⟩, ⟨ { ∀ind1 < 2.ref2 ˜donkey × ind1 }, { e2 } ⟩,
⟨ { ref3 ˜man! }, { e1 } ⟩, ⟨ { ∀ind2 < 1.ref4 ˜donkey! × ind2 }, { e3 } ⟩ }. Then

TDomain(SEnt, Sense, ∅) =

"∃x.(man(x) ∧ ∃Y.(∀y.(y∈Y ⇒ donkey(y)) ∧ |Y|=2 ∧

∃a.(donkey(a) ∧ ∃b.(donkey(b) ∧ a∈Y ∧ b∈Y ∧ "

+ TSense(Sense, { ⟨ e1, "x" ⟩, ⟨ e2, "Y" ⟩, ⟨ e3, "a" ⟩, ⟨ e4, "b" ⟩ }) + ")"

8.5. Sort Formation

TSort produces a string naming sorting relations on logical variables. The operation of TSort can be seen in the TDomain example above, in (for example) the application of the predicate "donkey" to the variable "a".

Specification:

For each sort symbol, s, in S, construct an expression consisting of the FOPC predicate named by s applied to the variable v. Conjoin all the expressions.

Definition:

If S = { s } then TSort(S, v) = `s + "(" + v + ")".
Otherwise, let s = Pick(S), then
TSort(S, v) = TSort({ s }, v) + " ∧ " + TSort(S \ { s }, v)

8.6. Partition Formation

TPartition yields a string naming relations between logical variables (specifically: ⊇ relations between the sets they denote). The relations are derived from the ordering ≫ on Ent. Its example is included in the TDomain example, above. TPartition is called as TPartition(e, v, C) where e is an entity token, v is the name of the FOPC variable which is the translation of e, and C is the set of existing ⟨entity token,variable name⟩ pairs from the discourse so far.

Specification:

If e is not after anything in ≫, there is no partition information.
Otherwise, let Y be the name of the variable associated with the entity token immediately before e under ≫. Then
If e is only bound to singular references, v ∈ Y holds.
Otherwise, Y ⊇ v holds.

Definition:

If $\neg\exists e'.(e' \in Ent \wedge e' \gg e)$ then TPartition(e, v, C) = "".

Otherwise, let $E = \{t : t \in Ent \setminus \{e\} \wedge t \gg e\}$.

Let $e' = Pick(E)$ s.t. $\neg\exists e''.(e'' \in E \setminus \{e'\} \wedge e' \gg e'') \wedge \langle e', v' \rangle \in C$; then

If all references bound to e are singular

TPartition(e, v, C) = "∧" + v + "∈" + v'

Otherwise

TPartition(e, v, C) = "∧" + v' + "⊇" + v.

8.7. Discourse Formation

The function TSense defines translation of the sequence of sentences in the discourse. It is defined inductively over the set of GRL sentences in the discourse state, and produces a string naming the conjunctions of their translations into FOPC. TSense is called by TDomain as TSense(Sense, C), where C is the set of ⟨entity token,variable name⟩ pairs created by TDomain.

Specification:

TSense yields the name of a conjunction of translations of expressions, given by calls to TExpr, in Sense.

Definition:

If $S = \{s\}$ and s is not context extended then

TSense(S, C) = ")" + TExpr(`PredOf(s), RefOf(s), C, "", "", "", "(") + ")".

If $S = \{coref(r_1, r_2)\}$ then

TSense(S, C) = ""

If $S = \{coref(r1, r2) \triangleright s'\}$ then

TSense(S, C) = TSense({ s' }, C).

If $S = \{s'' \triangleright s'\}$ then

TSense(S, C) = TSense({ s'' }, C) + " ∧ " + TSense({ s' }, C).

Otherwise, let s = First(S). Then

TSense(S, C) = TSense({ s }, C) + " ∧ " + TSense(S \ { s }, C).

Example:

The example of TSense (and of TExpr and TRef) is given as part of the larger example in Section 8.2. TSense constructs the conjunction of predicates which constitutes the main body of the expression.

8.8. Expression Formation

Specification:

The translation of each expression in Sense is a (possibly degenerate) disjunction, D_i, of FOPC expressions, D_{ij}. There is one disjunct (i) for each combination of individual entity tokens (j) appearing in each binding containing no indexed references.

Definition:

If $R = \varnothing$ then TExpr(P, R, C, Q, A, F, T) = Q + P + A + ")" + F.

Otherwise, let r = First(R). Then choose $\langle R',E\rangle$ such that $\langle R',E\rangle \in Bind \wedge r \in R'$.

If $\forall r'.(r' \in R' \Rightarrow r'$ is simple) then

TExpr(P, R, C, Q, A, F, T) =

TRef(P, R \ { r }, C, "$\vee$", { v : $\exists e.e \in E \wedge \langle e,v\rangle \in C$ }, Q, A + T, F, ",").

If $\exists r'.(r' \in R' \Rightarrow r'$ is indexed) then

TExpr(P, R, C, Q, A, F, T) =

TRef(P, R \ { r }, C, "$\wedge$", { v : $\exists e.e \in E \wedge \langle e,v\rangle \in C$ }, Q, A + T, F, ",").

8.9. Reference Formation

Specification:

Each D_{ij} is a conjunction of FOPC expressions, C_{ijk}. There is one conjunct for each combination of individual entity tokens appearing in each binding containing indexed references.

Each C_{ijk} is formed as follows:

Each quantifier $\forall I$ is replaced in each C_{ijk} by one or more $\forall x$ where X is the name of the FOPC variable associated by TDomain with the entity token bound to each reference associated with I and $x \in X$ holds.

The predicate symbol and modifers, P in C_{ijk} is replaced by the corresponding member of Predicates.

Each reference in each C_{ijk} is replaced by the variable associated by TDomain with the entity to which it is bound.

Definition:

If O = "$\vee$" then

If V = { v } then TRef(P, R, C, O, V, Q, A, F, T) = TExpr(P, R, C, Q, A + v, F, T).

Otherwise, let v = Pick(V); then

TRef(P, R, C, O, V, Q, A, F, T) =

TRef(P, R, C, O, { v }, Q, A, F, T) + "$\vee$" + TRef(P, R, C, O, V \ { v }, Q, A, F, T).

If O = "$\wedge$" then

If V = { v } then

Let u = gensym;

TRef(P, R, C, V, O, Q, A, F, T) =

TExpr(P, R, C, Q + "$\forall$" + u + ".(" + u + "$\in$"+ v + "$\Rightarrow$", A + u, ")" + F, T).

Otherwise, let v = Pick(V); then

Let u = gensym;

TRef(P, R, C, V, O, Q, A, F, T) =

TRef(P, R, C, { v }, O, Q , A, F, T) + "$\wedge$" + TRef(P, R, C, O, V \ { v }, Q, A, F, T).

Example:

The translations in figures 4a and b, above, show the full output of the translation function – *ie* the output of TWorld(WSort, Ent, Sense) for a particular possible translation of the input; as stated before, the translation function works separately on *each* of the parallel possible translations elaborated in the George system. The conjunction of universally quantified implications comes from TWorld itself; no claims are made regarding correctness, completeness, or even good sense, of the sort specifications – they are merely defined sufficiently for experimental purposes. The nested existentials are produced by TDomain, the sort predicates added immediately after the quantifier being added by TSort. The expressions conjoined to form the main body in each example are produced by TSense, their internal conjunctions and disjunctions being inserted by TExpr and TRef.

9. Summary

In this chapter, I have covered those parts of the George Representation Language which are relevant to the work presented here. I have introduced the following topics; discussion of the detail of those marked ★ has been deferred to later chapters.

1. Representing *references*, as primary objects in the language.
2. Representation of information about *sort*, *definiteness*, and *cardinality*.
★ 3. Quantification over sets by two kinds of *indexing*, *weak* and *strong*.
4. #-abstractions, which abstract sorts, as λ-abstractions abstract values.
5. Representing surface structure as far as subordinate clauses by *context extension*.
6. Representing entities in the discourse world by *entity tokens* denoting their nature, but not their specific form.
★ 7. Representing relationships between references and entity tokens with *bindings*.
★ 8. Representing cardinality relationships between sets by *bounding constraints*.

I have laid out a formal specification of the language, and given it a semantics by specifying an algorithm for translating the truth-conditional part of its content, given a context specified by George's output in terms of bindings and entity tokens, into First Order Predicate Calculus.

10. Afterword

We now have a formal language in which to work. Before we can begin to perform analysis of reference, we need a procedure for translating the input discourse into that language. The procedure used in the current implementation of the George system is presented in the next chapter.

Chapter 5
The George Parser

Abstract

The ideas motivating the design of the George Parser are restated in summary.

Basic categorial parsing is explained in detail, and it is argued that, by itself, categorial grammar is well-suited to, but not perfect for, deterministic incremental parsing. A means of overcoming this imperfection is introduced.

The structure of George's lexicon is defined.

The idea of adaptable representation is discussed, with a mechanism to extend the coverage of the simple categorial parser to a realistic level, with details of the ambiguous semantic forms introduced in earlier chapters.

1. Introduction

This chapter is a complete statement of the theory and implementation of the section of the George Parsing and DeReferencing System devoted to parsing natural language. The reader should be aware that, in making such a statement, certain assumptions have been made which are introduced and justified elsewhere in this document.

The most prevalent of the assumptions in question are as follows:

Introduced in Chapter 2:	Incremental reference evaluation can be useful; Incremental parsing is therefore appropriate;
Introduced in Chapter 3:	The "surface form" hypothesis – it is advantageous to use a representation rather closer to the surface form of an utterance than has been hitherto practised; The "ambiguity" hypothesis – it is possible and computationally useful explicitly to maintain ambiguity in unfinished translations.

No further justification of these is necessary, and therefore none will be attempted.

It is not immediately obvious that the content of this chapter is fundamental to the research presented here. Nevertheless, this is so. Without (some equivalent of) the new

ideas introduced here, adaptable, ambiguous representation, which is fundamental to my solution of the central set reference problem would be impossible.

2. A Suitable Formalism for Incremental Parsing

2.1. Introduction

In selecting the best formalism for incremental parsing, we must first consider what particular problems the incremental approach introduces. There are two main issues.

First, and fundamentally, we need to be able to create a self-contained representation of any left-complete partial sentence, in order that we may evaluate our semantics and reference word by word. It will also be convenient if there is a close correspondence between the syntactic and semantic representation of partial sentences, in that it is then easy to allow the same process to handle both syntactic and semantic composition. Then, the syntactic parsing operation can guide the semantic composition directly, which is an elegant state of affairs. (The correspondence will also prove useful later, when I define *protraction*.)

Second, we wish to reduce non-determinism as far as is possible. This is because, with the early processing involved in dealing with semantics and reference word by word, the computational cost of following dead end paths can be orders of magnitude larger than that of syntactic parsing alone. Non-determinism is, therefore, a much more significant issue in the incremental context than in other approaches, where it can sometimes be viewed as little more than an implementation detail. I will propose a new way of avoiding some non-determinism, by taking advantage of the adaptability of the George system.

One existing formalism which partly fulfills the first of these requirements is that of Categorial Grammar (CG). I use the term loosely here to include the recent developments in (*eg*) Combinatory Grammar in [Steedman, 1987] and Unification Categorial Grammar (UCG) in [Zeevat *et al*, 1986]. Later in this chapter I will explain a new development of CG, enhanced in ways comparable with these developments, which will allow adaptability to fulfil the second requirement, above.

2.2. Issues in Incremental Parsing

The choice of incremental parsing brings with it a number of practical issues, which I will cover in this chapter. The collection is, from a conventional linguistic point of view,

apparently arbitrary, because all the points of interest arise from clashes between the incremental parsing view and more traditional view of syntactic and semantic translation and composition of natural language, and thus are not motivated by, say, a desire to study a particular class of linguistic phenomenon. This begs a question of why the two views (*ie* conventional and early/incremental/adaptable) should be so much at odds in apparently arbitrary ways, and why the justifiable (I argued in Chapter 2) incremental view should disagree with traditionally received wisdom on the nature of human language understanding. Fortunately, for the purposes of this thesis, which is primarily of a practical nature, this can safely remain an open question.

The issues that I will cover, then, are all to do with fitting the conventional view of grammar (albeit expressed in a fairly up-to-date formalism) into an incremental style. For example, one issue is the problem of combining a (traditional) transitive verb phrase, which notionally already includes an object, with its subject, when the subject arrives at the input first. Another difficulty, apparently the converse, is the application of modifiers appearing after a phrase to that phrase when its (translated) structure has disappeared inside the translation of the sentence so far.

I will present two new mechanisms to solve these problems in what I suggest is a better way than before, on the grounds that it can lead to much more deterministic parsing than existing approaches to the same problems. In this chapter, I will show how these problem points are covered by the George system, and introduce two other issues, specifically head noun deletion and relative pronoun deletion which can be neatly dealt with under the new approach. The point of all this is that, although the collection of linguistic phenomena here may seem arbitrary, this is in fact not so – the problems arise as the result of mutual erosion between the edges of two apparently conflicting formalisms, and are therefore by nature rather irregular.

2.3. Categorial Grammars

The idea of categorial grammar began with [Adjukiewicz, 1935]. It has appeared in various forms in the literature (*eg* [Lambek, 1957]) and was recently revitalised in work (referenced above) by Steedman, Zeevat, Klein and Calder, and others at the Centre for Cognitive Science (formerly the School of Epistemics) in the University of Edinburgh, as well as by researchers in other institutions (*eg* [Dowty, 1985], [Moortgat, 1988]).

There are three reasons why CG is suitable for incremental parsing systems. Firstly, the modern developments of CG (mentioned above) are in principle capable of producing a

single valid translation for any meaningful sequence of words (though for our purposes, translation of any initial sub-sentence will be sufficient). This is the first requirement specified in Section 2.1 for George.

Further, the modern usage of CG is partly motivated by the fact that it is easy to construct a compositional semantics whose functional structure corresponds closely (or even exactly) with the functional structure of the grammar. This also is useful for strictly incremental parsing.

The third reason is that CG facilitates the maintenance of implicit ambiguity in the representation of translations. This point will be covered in Section 7; the argument will require information not yet presented.

In general, then, Categorial Grammar is an ideal formalism for an incremental system like George. It will be necessary, though, to adapt the ideas of basic CG (as in [Lambek, 1957]) to iron out a few significant problems (the same problems which are covered in the extensions from CG to CCG and UCG). These will be explained later in this chapter.

2.4. Categorial Notation and Type Raising

2.4.1. Two Notations for Categorial Grammars

In the literature on Categorial Grammars we can find two slightly (but significantly) different notations.

In [Lambek, 1958], as with all categorial notations, atomic syntactic categories are chosen from some small set (*eg* { s np n } denoting { sentence noun-phrase noun } respectively), and functions over categories are defined in terms of the categories of the argument and result. In particular, a function which applies backwards to its argument is written with the argument category first, and the result category second, separated by a backward oblique. For example, an intransitive verb (which applies backwards to an np to give an s) has functional category np\s. Forward application is written the other way round, with a forward oblique; so the category of a determiner (which applies forwards to a noun to give a noun phrase) is written np/n.

A simple set of basic combination rules for a grammar allowing *forward* and *backward application* (hence these are *application rules*) would look like 47a and b respectively (where X and Y range over basic and functional categories):

X/Y + Y	→	X	47a
Y + Y\X	→	X	47b

The advantage of this notation is that it maintains two separate argument lists for its functions, one for each direction of combination: the orders of the categories on either side of the main result category are exactly the orders in which they combine, forwards or backwards, with their arguments. 48 is an example of the combination of subject noun phrase with a mono-transitive verb.

np + np\s/np	–application→	s/np	48

According to Lambek's rules, above, we could equally well perform the combination of the verb with a post-fixed object noun phrase, thus:

np\s/np + np	–application→	np\s	49

The syntax of the notation reflects an ability to apply the function to its arguments in either order, the respective orderings of the arguments forwards and backwards being independent.

In later work on Categorial Grammar (*eg* [Steedman, 1985]), a slightly different notation has become common – which gives rise to a different set of combination rules for our example grammar. The rules are given in 50; compare with those in 47, which define exactly the same behaviour for Lambek's notation. Upper case letters in all the rule statements denote variables (ranging over categories, both basic and functional).

X/Y + Y	→	X	
Y + X\Y	→	X	50

The notational difference is simply that a function applying backwards to a member of category Y to give a member of category X is written X\Y instead of Y\X. This has further reaching consequences that one might at first realise.

(For completeness at this stage, I must also introduce Steedman's *forward composition* operation, which is defined as in 51. It will become clear later that we do not need this in George, but it will be useful for the moment for the purposes of general discussion of CG.

X/Y + Y/Z	→	X/Z	51)

Now, consider again the example in 48. The verb category, in Steedman's notation, is written s\np/np. Some authors go further, and write (s\np)/np, to ensure and emphasise that the syntax of the linguistic entity (in this case a mono-transitive verb) corresponds

structurally with its semantics – it is often useful to think of the semantics of a mono-transitive verb as a function applying to an object noun phrase, and yielding an intransitive verb. The result of this notational difference is that the two lists of arguments (forward and backward) which were neatly separated in Lambek's notation are now interleaved, with the result that, when we attempt to perform the operation

* np + (s\np)/np –application→ s/np 52

we are prevented from so doing by the requirement implicit in the notation first to combine forwards with an np.

This presents us with a rather more fundamental problem than how simply to "unwrap" the brackets, as it were, in a syntactic category in the Steedman notation. Recall from Chapter 3 that the problem has arisen in the first place from a desire to make the structure of the syntax of our grammar correspond with that of the semantics associated with it. This desire is worthwhile and justifiable in a context where we wish to perform composition of semantics, as well as syntax, word by word; and this, for the purposes of George, we must do – otherwise, we will be unable to perform word by word reference evaluation, because that evaluation depends on the semantics of noun phrases.

We are left with several solutions to our problem. Either we must find a means of expressing semantic functions which will allow their representation in a way functionally isomorphic with the Lambek notation of the corresponding syntax (but Lambek does not discuss semantic representation, except in viewing syntax as a very coarse grained semantics), or we must adapt parsing of the Steedman notation in such a way that we can still make combinations like that in 52 incrementally. (A further possibility is the approach of situation semantics (see *eg* [Barwise & Perry, 1983]), where global syntactic rôles are used instead of abstraction).

George takes the option of adapting the Steedman notation. The point is that there exists already a standard functional notation system (the λ-notation and its associated reduction and composition rules), and that the use among current researchers of the Steedman notation for CG is widespread. There seems little point in designing a new, non-standard semantic notation to support the use of the archaic syntactic one used by Lambek.

So, having made this decision, how do we approach this direction and ordering problem associated with the Steedman notation?

2.4.2. Type Raising

Let us first consider Steedman's own solution (*eg* [Steedman, 1985]). Steedman suggests that one way of making the failed combination in expression 52 work would be, instead of attempting to change the function, to change the argument. We cannot justifiably change the way the argument combines with the function as it stands; this would amount to parsing the words in reverse order. We can, however, modify the noun phrase in such a way that it becomes a function which can compose with the verb as we require it to, and in the correct (forwards) order. This operation, which is applicable only to certain categories (*eg* head noun, nominal), is called *type raising*. The left-most term in 53 is a type-raised noun phrase.

$$s/(s\backslash np) \;+\; (s\backslash np)/np \;\text{–composition→}\; s/np \qquad 53$$

We can now compose the rest of the sentence in the conventional way for CG.

However, application of the type raising operation begs a very important question: what about the semantics – consideration of which, after all, led to the emergence of this problem in the first place? In answer, we can propose a semantic change correspondent with the syntactic one.

Suppose that $\lambda x.\lambda y.\Phi(x,y)$ is the semantic expression representing the mono-transitive verb (category s\np/np) above, and τ is the translation of the head noun phrase. Then by simple substitution, $\lambda x.\Phi(x,\tau)$ is the translation of the resulting item of category s/np. The transformation we need to apply to τ to get this behaviour is

$$\tau \;\text{–type\_raising→}\; \lambda F.F(\tau) \qquad 54$$

which is clearly a general transformation, because it does not depend on τ. Now, we can compose the semantic functions, as in 55, to give a correct translation. (The composition operation used here is the usual one for composition of λ expressions. The square brackets are used merely to aid readability.)

$$\begin{array}{lll} \lambda F.F(\tau) + \lambda x.\lambda y.\Phi(x,y) & \text{–composition→} & \lambda x.([\lambda F.F(\tau)].[\lambda y.\Phi(x,y)]) \\ & \text{–reduction→} & \lambda x.([\lambda y.\Phi(x,y)](\tau)) \\ & \text{–reduction→} & \lambda x.\Phi(x,\tau) \end{array} \qquad 55$$

The essential principle being embodied here is more than a simple textual manipulation of the λ-notation. What we are representing is a change of functional order in the translation of the noun phrase, from a mapping from $\varnothing$ (empty set) to {constants} (*ie* a

constant function) to a mapping from {functions of arity two} to {functions of arity one} – hence the term "type raising".

In isolation, type raising certainly seems to be an effective solution to our current problem. However, in some ways it seems more than a little *ad hoc*.

First it is not a general solution: to use it correctly, we must make a rule that only head noun phrases, and some other categories (*eg* nominals) may be type-raised. The justification for this depends solely (and therefore, surely, inadequately) on the fact that these categories are the only things which need to be type-raised to facilitate incremental parsing. Further, type-raising does not fit in to a more general solution to other issues, such as the perennial problem (which is addressed in [Pareschi & Steedman, 1988] and [Pareschi, 1988]) of backwards application of modifiers later in a sentence, when the structure of the exsiting composition has been buried in the higher-level category (usually s/X for some category X).

Secondly, there is no formal rationale at all for the complicated functional form of the type-raised entity: in a context where the normal representation of (the translation of) a noun phrase is a constant, such complexity, unjustified, in an expression is surely out of place. Worse, there is no general statement of meaning for the semantic expression corresponding with a type raised object (aside from saying that it is the same as that of the object before it was type raised). It will become clear in Chapter 6 that this is undesirable, from the points of view of both elegance and computational efficiency, when we attempt to evaluate the reference of partial sentences.

Another function of type-raising in some theories is to allow production of different quantifier scopes. Here, the notion of changing the order of a function fits much more comfortably, because quantifier scoping exactly corresponds with the order of combination of lexical items. However, for the adaptable representation I propose here, it is highly undesirable that quantifier scope should be denoted by bracketting in expressions, since it is hard to imagine a representation which intuitively represents two differently scoped versions of the same quantified predications at once, unless it is without any scope at all. What is more, [VanLehn, 1978] (cited by [Sidner, 1983])

> "reviews the major theories of scoping phenomena and concludes that disambiguation of quantifier scope does not seem to take place during parsing or semantic interpretation of a sentence and that the determination of quantifier scope is the result of other linguistic processing."

George's approach is exactly in line with this suggestion.

3. The George Parser

3.1. Introduction

Having raised these questions, I will now begin to supply some answers. This section explains the operation of the George parser, with particular emphasis on the issues of incrementality, adaptability and the operation which George uses instead of type-raising. Before dealing with the declarative statements of the rules and principles used in the George system, it will be useful to discuss an overview of the process model which George follows. The procedural model is presented below as such, the purely declarative rules and principles used within the procedures being stated subsequently.

3.2. George's Process Model

Figure 5 represents the top level of George's parsing process model. The terms used in the figure were defined in Chapter 4. The single-line boxes denote data structures; the double ones denote processes. The arrows represent dataflow. The diagram represents a circular

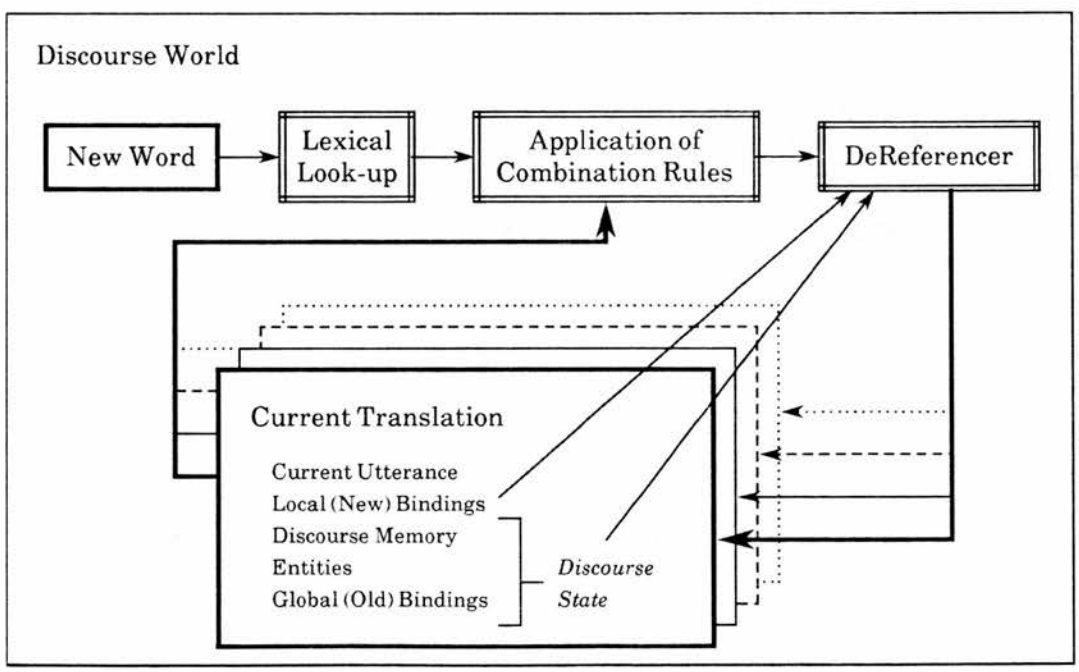


Figure 5: *George's Process model*

process. Each time a new word arrives at the input, the lexicon is consulted, and all possible translations of the word are found (by deduction if necessary – see Section 3.6). An

attempt is made to combine each of these translations with the "current utterance" part of each possible translation existing at the time. If a particular combination fails (under rules specified below) the parser proceeds no further along that branch of the search tree.

If a combination succeeds, then the result (again, defined below) is passed to the DeReferencer (see Chapter 6), in the form of a set of George references, extracted directly from the current translation by a simple textual operation. Each time such a set is received by the DeReferencer, any information in it about references which already exist replaces that already known. This is a requirement of adaptability – if we wish our GRL expressions to be adaptable, the copies of them elsewhere in the system must also be adaptable, otherwise inconsistency could arise.

Recall from Chapter 3 that the current (partial) utterance is not included in the discourse state until it can be viewed as a "complete utterance". This is detected, once an end-of-sentence flag has been received, by matching the syntactic category against certain distinguished syntactic labels (sentence, noun phrase, and prepositional phrase currently, though the content of the full set is an open question). For the purposes of discussion here, I will cover only full sentences, since detailed work on the other cases has not been carried out.

In the event that one of the distinguished categories has been obtained in a particular translation, the erstwhile current utterance and its bindings are appended to the discourse memory and the global bindings, respectively, and the current utterance and local bindings are reset to empty. It is safe to update the Entity Tokens database incrementally throughout the process because the DeReferencer ensures that the list is consistent (*ie* it does not contain any entities to which no reference has been made), so it does not matter if entities are created prematurely as candidates for reference. Therefore, by this stage, the entities list will have been updated with all the relevant information. (All this will be covered in Chapter 6.) So now we are ready to accept the first word of the next utterance and to start the process again.

If we have not reached a distinguished category, we replace any out-of-date local bindings with the new ones produced by the DeReferencer and add the current utterance to the current translation (the entities already having been updated incrementally by the DeReferencer, as above), and wait to start the process again for the next word of the sentence.

At this top level, the model is very simple and uniform. As we move deeper into the detail, below, George becomes more complicated. Uniformity, though, has been a major design

constraint – it will become clear below that it has been maintained into the full depth of the George system.

3.3. Combining Partial Utterances in George

Some basic operations for combining partial utterances in GRL (*eg* λ-reduction) were defined in Chapter 4. At this higher level of the process model we need a means of deciding which one(s) of these primitives to apply, and how to apply it(them).

This information is specified by a (small) number of *combination rules*; in fact, those already specified in 50 form a special subset, the *basic combination rules*. The combination rules are of the following general form.

$$X_1{:}\Sigma_1 + X_2{:}\Sigma_2 \rightarrow X_3{:}\Sigma_3 \quad \Leftarrow \quad \textit{conditions} \qquad 56$$

where X_i and Σ_i are variables ranging over categories and expressions in GRL respectively and : associates pairs of these together. *conditions* are an arbitrary logical formula, which will usually be empty (in such cases, the ⇐ operator will be omitted). While I have not specified any restrictions on the form of *conditions* it seems likely that in general either they will be a fairly simple predication of properties of (parts of) the combinands and/or the result, as in the termination rule, explained below, or they will be recursive calls of the combination rules like those which admit protraction and coercion here (again, explained below). Thus, the notion of implication here is procedural, as, for example, in Prolog.

Given two items which we wish to combine, say

$$X_1{:}\Sigma_1 \; + \; X_2{:}\Sigma_2 \qquad 57$$

we attempt to unify (using first-order term unification) with the left hand side (that is, the part to the left of the first →) of all the available rules. In general, the semantic parts of the rules will be unconnected at this stage; but they and (more usually) the two syntactic categories may contain references to common variables. This allows us neatly to check the syntactic compatibility of the combinands, and thence their compatibility under whichever operation the rule specifies, by simple unification. Next, we attempt to prove the conditions, if there are any. If these succeed, unification gives us a result. For example, if we wish to combine the following function and argument

$$\text{np/n} : \text{refl\#refl} \; + \; \text{n} : \text{man} \qquad 58$$

we could match it with the following left-hand-side (using upper case characters to represent variables).

$$X/Y : \Sigma_1 \ + \ Y : \Sigma_2 \qquad 59$$

X unifies with the np, and Y with both the n's. Σ_1 and Σ_2 are unrestricted in unifying with the respective semantic expressions, because they are named distinctly. Now, if the category of the second combinand had been, say, np, this unification would have failed, and therefore the rule would not have matched, which is the behaviour we want. It can be seen throughout this discussion that this approach allows us to specify our rules in a way exactly equivalent to the *de facto* standard (Steedman) notation for categorial rules.

Once we have performed the unification to show the syntactic correctness of the combination, we can check the well-typedness of the semantic expressions; for example, we can check that a #-expression is indeed being applied to a discourse world sort (as required by the definitions in Chapter 4). Once the well-typedness has been shown, we can go ahead and do the combination, which will normally involve a reduction step and an evaluation step (again as defined in Chapter 4), and obtain our result.

3.4. Basic Combination Rules

The basic mechanism of the George Parser is built on a simple shift-reduce parser, borrowed from [Haddock, 1989], though now it is altered almost beyond recognition. The parser is a general incremental parser for categorial grammars, because the user is able to specify not only the lexicon, but also the combination rules, independently from the parsing mechanism. It is in truth not clear that one would necessarily wish to be able to alter the combination rules between different grammar implementations; ideally, we would like to think that the set of rules, once expressed, is universal and immutable. However, experience (throughout, for example, Steedman's work on CG) has shown that the correct set of rules, if such a one exists, has not yet been found; each variant on basic CG, designed to solve a particular problem, seems to require a different set.

Therefore, in George, the grammar writer has the option of changing the combination rules. The stated intention is that those already supplied (and detailed below) should be a necessary and, possibly, sufficient set for parsing English in the George paradigm, though no claims are made here for sufficiency.

George's combination rules are expressed as schemata specifying the form of the application and composition operations. Possibly because of the small coverage of English required for the study of the particular linguistic phenomena with which we are concerned here, George currently requires only two basic combination rules – the same two application rules used in the foregoing example. The logical specification of the basic rules is shown in full in 60.

$$\begin{array}{lcll} X/Y{:}\Phi + Y{:}\alpha & \rightarrow & X{:}\Phi(\alpha) & \\ Y{:}\alpha + X\backslash Y{:}\Phi & \rightarrow & X{:}\Phi(\alpha) & 60 \end{array}$$

Roman characters denote syntactic categories (basic or functional); Greek characters denote expressions in GRL, upper case being functions, and lower case atomic expressions (which may be functions treated as such). Both kinds of character represent variables in the rules, which are associated with the real functions and arguments by first-order term unification, which enables the initial syntactic compatibility check between function and arguments mentioned above (*cf* UCG in [Zeevat *et al*, 1985]).

It is worth emphasising again the nature of George rule definitions. Any of George's rules as stated in this chapter may be made conditional upon an arbitrary logical formula, in order, for example, to express preconditions on the rules. In particular, it will be seen later that this is useful in specifying complicated rules recursively in terms of the more basic ones. Such rules are written thus:

$$\textit{Rule} \quad \Leftarrow \quad \textit{Formula} \qquad 61$$

Formula will often contain variables instantiated in *Rule*. It is important to understand, though, that the conditional operator, $\Leftarrow$, is used here procedurally, and not in the classical logical sense. In other words, the statement above indicates that attempted application of *Rule* may only succeed if *Formula* is true, given the variable instantiations caused by matching with *Rule*, in the backward chaining sense (of, for example, Prolog). It does not indicate that, if *Formula* is true, we may assume the applicability of *Rule*. *Formula* is evaluated as part of the attempt to determine whether *Rule* is applicable; the result it produces in binding its variables constitutes the construction which justifies the application of the rule. In the George grammar as it stands, only one basic rule has such a conditional, and the others use the conditional just to introduce recursion; but implementers of other grammars would be able to include arbitrary conditionals in any rule, if they so desired.

In accordance with the specifications given in Chapter 4, application and composition (the latter for explanation only) are only defined for particular combinations of entities in GRL. #-abstractions apply to sorts and nothing else; λ-abstractions compose with λ-abstractions or with #-abstractions (but not *vice versa*). Because of this strict typing of GRL, when we are attempting to select a rule, the success of that selection depends not only on the rule selection by unification described above, but also on the successful reduction of any function applications or compositions incurred, which involves checking the compatibility of the George types.

Since we are parsing strictly incrementally, the right hand argument to one of the combination rules will always be the translation of a single word, while the left hand will always be a left-complete partial sentence. Because of this, it is convenient and elegant to include the identity combination rule 62, where ∅:∅ denotes an empty translation. The use of this rule is demonstrated in Section 3.7.

$$\varnothing : \varnothing + X{:}\Phi \qquad \rightarrow \qquad X{:}\Phi \qquad 62$$

This rule allows us to parse uniformly from the first word with no special case built into the parser for the initial word, thus maintaining the parser's generality. Similarly, we can introduce a rule for termination, using ∅:∅ to represent the terminating symbol. This rule, shown in 63, uses a conditional formula to ensure that the syntactic category of the complete utterance is one of the distinguished categories.

$$X{:}\Phi + \varnothing : \varnothing \rightarrow X{:}\Phi \qquad \Leftarrow \qquad \textit{distinguished}(\,X\,) \qquad 63$$

George needs this rule because (if the George Parser is to be fully general) it may be necessary to apply certain operations – coercions, details of which will be given later – to a partial sentence; this applies to the result of the very last combination in a sentence, too. Now, this rule, in conjunction with that applying coercions, enables an adaptable, unfinished sentence to be coerced to a complete one if this is possible, again without building such a facility into the parser; and we do not want to break the existing division between parser and grammar rules by building the rules into the parsing system itself. The condition that the result be one of the distinguished "complete" categories mentioned before is conveniently applied here as a condition on the rule; again, the grammar implementer is free to change this condition without reference to the parsing mechanism.

This constitutes all the basic (*ie* non-recursive) parsing rules.

3.5. Parsing Primitives

Having defined our basic rules, we next need a parsing engine with which to drive them. George has two primitive operations for this purpose, *shift* and *reduce*. Recall from the description of George's process model in Section 3.2 that these two are called by the main program in strict alternation: it is only possible to *shift* the set of translations of one new word at a time, because the stack of [Haddock, 1989]'s parser has been replaced with a single fixed storage location.

shift is defined, given a word W, to move a set T in to the "current words" slot of the parser in a the way shown in 64 (The := relation connects words with their lexical entries – details in Section 3.6.).

$$T = \{ t : (W := t) \text{ exists in the lexicon} \} \qquad 64$$

The *reduce* operation is rather more complicated, and requires a little more explanation. Given a discourse world, as defined in Chapter 4, a current translation is a quintuple, $\langle s, L, D, E, B \rangle$, where s is the translation in GRL achieved so far of the current sentence, L is a set of bindings between references in s and entities in E, D is a set of complete translations of preceding sentences, E is a set of entity tokens arising from the discourse, and B is a set of bindings between references in D and entities in E. $\langle D, E, B \rangle$ forms the discourse state. Then we can say:

A set of discourse states, N, may be produced by *reducing*
a set of discourse states, O, with a non-empty set of translations T

$$\Leftrightarrow \quad N = \{ \langle s', L', D, E', B' \rangle : \exists t \in T, \langle s, L, D, E, B \rangle \in O. \; (s + t \rightarrow s' \wedge \textit{reference}(\langle s', L', D, E', B' \rangle, L, E, B)) \} \qquad 65$$

reference is a relation between one current translation, and the three elements of another ($\langle L, E, B \rangle$) which define reference. For the moment, it is enough to say that *reference* is true if the reference of the new structure is in some sense consistent in itself, and consistent with the reference already existing in the preceding discourse. This loose notion of reference will be formalised in Chapters 6 and 7.

If the set N in the definition above is empty – *ie* if all possible syntactic combinations have failed – then the whole parse is deemed to have failed.

3.6. The Lexicon

The final component of this snapshot of the basic parser is the lexicon. George's lexicon is almost trivially simple. It specifies an association between words and syntax/semantics pairs of the form *word* := *syntactic category*:*semantic expression*, some examples of which are shown in 66. The notation is as defined in Chapter 4. (though the program does not use exactly this notation – its version is rather less readable). The references, variables, and indices ($refN_i$, $varN_i$, and $indN_i$, respectively) are made unique by choosing a unique integer value for N_i each time the lexicon is consulted.

$$\begin{aligned}
&\text{the} && := \text{np(plur, __)/n(plur) : } refN_1 \# \forall indN_2.refN_1 ! \times indN_2 \\
&\text{the} && := \text{np(sing, __)/n(sing) : } refN_1 \# refN_1 ! \\
&\text{man} && := \text{n(sing) : man} \\
&\text{eats} && := \text{s\np(sing, nom) : } \lambda varN_1.[\text{eats},varN_1]
\end{aligned} \qquad 66$$

It has also been possible to generalise entries for (*eg*) nouns which form plurals by the addition of "s" to the singular, by the inclusion of lexical redundancy (or deductive) rules which relate the translation of a set of words from others which are directly defined, as shown in 67. The detail of the *forms_plural_with_s* predicate is irrelevant here; the entailment operator $\Leftarrow$ is the same as that appearing in the combination rules.

$$\begin{aligned}
\text{Word} := \text{n(plur):Sort} \Leftarrow\ & \textit{forms_plural_with_s}(\text{ Word }) \\
\wedge\ & \text{Singular + "s"} \xrightarrow{\text{concatenation}} \text{Word} \\
\wedge\ & \text{Singular} := \text{n(sing):Sort}
\end{aligned} \qquad 67$$

When we try to look up a word ending in "s", this rule checks to see if it is a plural of a word which forms its plural with "s". If so, the lexical entry is derived from the lexical entry for the singular form (which happens to be an easy transformation in George, and so this rule is worthwhile).

As is the case universally in George, pattern matching in the lexicon is done by first-order term unification; matching an instance (in the case of the lexicon, an English word) against a set of patterns yields an exhaustive set of possible matches (in the case of the lexicon, translations). There is no implicit or explicit ordering on the items contained in the lexicon (nor, indeed, on the combination rules). An implication of this is that preferences may not be attached to some translations over others by ordering; if such preferences are to be expressed, the George Representation Language must be extended to

include their expression, which is certainly better (from a grammar designer's point of view) than preferences implied by ordering , because the mechanism is explicit.

Notice also that a further addition has been made in 66 to the elementary categorial syntax described earlier in the form of simple feature structures appended to the category names np and n. These features allow us to perform agreement checks just using first-order term unification (which we need anyway for our basic syntactic agreement checking). For example, n has one feature, which may be *sing*ular or *plur*al – other kinds of noun are not covered in George. np has two features, one for number as with n, and one for case, which may take the values *nom*inative or *acc*usative, for agreement with verb argument positions. The Prolog notation, _, for the unnamed variable or "Don't Care" is often used in these feature specifications.

3.7. A Simple Example of George Parsing

As an example of how this foundation level of the parser works, then, let us consider how the sentence "The man eats." would be parsed with the simple rule set and lexicon given in 50 and 65, respectively. Suppose, for simplicity, that this is the first sentence of the discourse; for the same reason, I will not mention reference here. Recall that, at this stage in our discussion, I am using only the most basic of George's combination rules.

Remember that, in the following explanation, wherever a combination rule is applied, the associated evaluation, as defined in Chapter 4 is also performed. Thus, an expression $\Phi(\alpha)$ will be replaced by Ψ, where Ψ is the result of carrying out *evaluation* upon $\Phi(\alpha)$, as defined in Chapter 4..

To start, we have the initial set of one empty discourse state:

$$\{\langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \rangle\} \qquad 68$$

The empty set structure in the current sentence (leftmost) position expresses the fact that there is no preceding word in this sentence. The other sets (new bindings, discourse memory, entities, and old bindings) are empty, of course, because this is the beginning of the discourse.

We attempt to *shift* the word "The". The result is the set of lexical entries

$$\begin{aligned}\{&\mathrm{np}(\mathrm{plur}, \_)/\mathrm{n}(\mathrm{plur}) : \mathrm{ref1}\#\forall \mathrm{ind1}.\mathrm{ref1}! \times \mathrm{ind1},\\ &\mathrm{np}(\mathrm{sing}, \_)/\mathrm{n}(\mathrm{sing}) : \mathrm{ref1}\#\mathrm{ref1}!\}\end{aligned} \qquad 69$$

which we combine with the initial (empty) state, using the identity combination in 61.

So, when we first perform the *reduce* operation using the identity combination, we obtain a set of two discourse structures, like this (leaving discussion of the content of L_i and E for Chapter 6):

$$\{\langle \text{np(plur, \_\_)/n(plur) : ref1\#}\forall\text{ind1.ref1!}\times\text{ind1, } L_1, \varnothing, E, \varnothing\rangle,$$
$$\langle \text{np(sing, \_\_)/n(sing) : ref1\#ref1!, } L_2, \varnothing, E, \varnothing\rangle\} \quad 70$$

Next, we must *shift* again, because of the strict alternation by the parser between *shift* and *reduce*, by which George enforces incrementality. Since there is only one translation of "man" available, the set of new lexical entries is simply

$$\{\text{n(sing) : man}\} \quad 71$$

When we attempt to *reduce* this with the current translation, only the forward application rule – the first rule in 60 – has a left hand side whose respective components are potentially unifiable at the functor level with both the function, from 70, and the argument, from 71. However, because of the requirement also to unify the number feature associated with the argument categories, only one of these possibilities will be realised, to give a singleton set of discourse states:

$$\{\langle \text{np(sing, \_\_) : ref1\#ref1}^{-}\text{man!, } L_2, \varnothing, E, \varnothing\rangle\} \quad 72$$

Now we can *shift* the translation of "eats" and *reduce* again to give:

$$\{\langle \text{s:[eats,ref1}^{-}\text{man!]}, L_2, \varnothing, E, \varnothing\rangle\} \quad 73$$

Finally, the appearance of a full-stop at the input indicates the end of a sentence (representing an ending inflection in speech, if appropriate). This causes introduction of the end of utterance translation, $\varnothing$:$\varnothing$. The only rule with which this is unifiable is the "end of sentence" rule shown in 62. In accordance with that rule, the category of the left hand combinand is examined, to see if it is one of the categories distinguished as constituting a complete utterance. Since it is (recall that the distinguished categories are { s np pp }), the combination succeeds, and the sentence is complete. This causes the transfer of the current sentence to the discourse memory (and the corresponding operation on references and entity tokens), yielding the singleton set of discourse structures shown in 74. (Recall that the ordering of the discourse memory – as defined in Chapter 4, the third element of each tuple – is significant.)

$$\{\langle \varnothing, \varnothing, \{s : [\text{eats,ref1}^{\frown}\text{man!}]\}, E, L_2 \rangle\} \qquad 74$$

4. The "Type Raising" Problem and Protraction

4.1. The Protraction Operation

While the above example shows how the most basic elements of the George parser fit together, it bypasses the difficulty of directional combination in incremental parsing, because the verb involved is intransitive, and so has only one argument.

To remove with that difficulty, I have introduced a new operation, called *protraction*, into the parser. Because of the generality of the *shift* and *reduce* specifications, I can do so elegantly by adding a recursive element to the existing combination rules – the two new rules shown in 75. υ and r are variables ranging respectively over the sets of symbols Var and Ref, introduced in Chapter 4. Recall that the $\Leftarrow$ symbol denotes procedural dependency; that is to say, the left-hand side of an expression containing $\Leftarrow$ is dependent on the back-chained execution of the right hand side, as in Prolog. Thus, these rules express not an ability to prepend arbitrary abstractions to syntactic categories (as one might suppose, reading from right to left), but a dependency of the result of the left-hand rule application on that of the right-hand one.

$$\begin{aligned} &T : \Phi + V/U : \lambda\upsilon.\Psi \rightarrow Y/U : \lambda\upsilon.\Xi \quad \Leftarrow \quad T : \Phi + V : \Psi \rightarrow Y : \Xi \\ &T : \Phi + V/U : r\#\Psi \rightarrow Y/U : r\#\Xi \quad \Leftarrow \quad T : \Phi + V : \Psi \rightarrow Y : \Xi \end{aligned} \qquad 75$$

These rules, defining λ- and *#-protraction*, respectively, may be expressed by one statement in English:

> Any item V/U:Σ, where Σ is an abstracted expression, may be combined with another item T:Φ if T:Φ+V:Σ' → Y:Σ", where Σ' is produced by temporarily removing the outermost abstraction operator and its associated bound symbol from Σ and treating any occurrences of that bound symbol as constant while performing the combination. The final result of the combination is formed by replacing the abstraction symbol in front of the intermediate result, Σ".

In procedural terms, the program implements this as three distinct phases – stripping of the abstraction operator, the intermediate combination, and the replacement of the abstract operator – implicit in the recursive execution of the defining rules above. It is fundamental to the operation of the George system that these three phases happen within

the analysis of one word. Thus, no information is stored outside the translation between words; so we always have a self-contained partial translation.

This operation is called protraction because it is almost the reverse of abstraction, except that we do not have a value with which to replace the bound symbol – if we did, we would be doing reduction. Instead, we are assuming that there will at some stage be a substitution for the bound symbol; and we are projecting ahead to that time, using the protracted variable or reference as a token for the value which it will eventually take. (This might be viewed as the extreme of adaptability – the protracted symbol is being used to stand for any GRL entity which might later be subsituted for it.)

Under this definition, one consequence of the protraction operation might be that if the protracted symbol appeared in the expression being combined into the protracted expression, we could get incorrect translations when that protracted variable was eventually combined with its own argument (because of something akin to variable capture). However, this can never happen, because the symbols being protracted are always unique constants (guaranteed so because they are generated individually by the lexical lookup giving rise to them); they are therefore certain to be mutually unrelated, and so it is safe to manipulate them independently without fear of creating contradictions.

I have included here only the rules which allow protraction to apply to the second combinand in a combination. It seems likely that a similar operation might be required for the first combinand, if we wish, for example, to allow combination of categories usually composed under Steedman's Backward Crossed Composition rule, which I will discuss further in Section 4.3. For the existing limited linguistic coverage of George, this form of composition is not necessary.

4.2. An Example of the Use of Protraction

To show how protraction works, I will extend the previous parsing example to one which requires protraction. Consider the sentence "The man eats the ham.". First, we need two new lexical entries:

ham := n(sing) : ham

eats := s\np(sing, nom)/np(__, acc) : $\lambda varN_1.\lambda varN_2.[eats,varN_1,varN_2]$ 76

With the λ-protraction rule specified in 75, we can proceed as follows. The assembly of the first noun phrase is as before (in 69 to 72). Ignoring for this example the ambiguity which now arises between the in- and mono-transitive readings of "eats" (which is anyway

represented at a higher level by replication of George processes), we now *shift* the word "eats". This leaves us trying to find a rule to allow the combination in 77.

np(sing, __):refl~man! + s\np(sing, nom)/np(__, acc):λvar1.λvar2.[eats,var1,var2] 77

Now, the basic combination rules given in 60 will not unify with this combination, but when the λ-protraction rule in 75 is applied, it unifies successfully, with this unifier:

{ T = np(sing, __), V = s\np(sing, nom), U = np(__, acc),
Φ = refl~man!, Ψ = λvar2.[eats,var1,var2], υ = var1 } 78

and initiates a recursive call of the combination operation on the combination in 79. Recall that, when in the recursively called environment, we view var1 as a constant or at least an externally bound variable; the context of evaluation is otherwise unchanged.

np(sing, __) : refl~man! + s\np(sing, nom) : λvar2.[eats,var1,var2] 79

We can make this application with the basic combination rules.

s : [eats,var1,refl~man!] 80

We have now completed the recursive call, so on returning we effectively replace the abstraction operator which was temporarily removed to give 81:

s/np(__, acc) : λvar1.[eats,var1,refl~man!] 81

To perform the next combination, with an item of category np/n, we will need to use the #-protraction rule; the form is identical to the above. Thenceforward, we can proceed using just the basic rules, in the same way as before.

4.3. Some Comments about Protraction

Having made this definition of protraction, there are some important points to note.

4.3.1. The Structure of Syntax and Semantics

Protraction is entirely dependent on the isomorphism between syntactic and semantic functions assumed by George. This, however, is not a disadvantage; the correspondence is a good thing, if only because it makes lexical entries easier to understand and design. Also, there seems to be a general intuition amongst linguists that such an isomorphism is

in some sense "correct". It is also dependent on the first-order nature of GRL evaluation and on the use of λ-(or β-)reduction as merely a notation of textual substitution, because these mean that semantic function application maintains the structure of components in the result of their combination.

4.3.2. Stacks and Non-Determinism

Protraction may be considered as enabling exactly the same effects as a stack in a conventional shift/reduce parser, and as the stack simulation hidden in the semantic transformation of type-raising. Indeed, the recursive nature of the definition implies the presence of a stack behind the protraction operation. However, the actual use of the stack in protraction is fundamentally different, in that un-combinable items are not stored for re-trial after reading more words from the input. Instead, we are, as it were, immediately *pulling the status of our translation forwards* (hence *pro-traction*) to a situation, if such a one exists, where those words will have been read and the current combination will be possible. Therefore, we are always able to find a fully evaluated expression (if one exists) representing the syntax and semantics of a given partial utterance. Protraction allows us to do this without compromising our position of strictly incremental parsing, because we never leave items stacked for longer than it takes to combine any single word. What is more, because protraction is demand-driven (*ie* it only arises in response to the attempt to combine two given categories) is can never introduce unmotivated non-determinism, as (*eg*) type-raising does frequently.

The upshot of this is that all the information about a translation can always be encoded in that translation itself, rather than being partly off-loaded into stack storage. Therefore, at any time during the parse of a sentence, we are able to treat the current analysis as a complete translation of the part of the utterance so far covered. This is a fundamental requirement for a system which parses strictly incrementally.

4.3.3. Protraction and Composition

With the exception of one special case, which will be detailed in Section 6, and which is, anyway, resultant from a particular idiosyncrasy of the representation being used here, the protraction rules subsume the action of the more traditional composition operation and render it obsolete. We therefore no longer need to consider it as part of our grammar specification, improving the simplicity and uniformity of our rule system. The proof of this for the syntax of the usual Forward Composition and Generalised Forward Composition

rules follows – proof of syntactic equivalence implies semantic equivalence because the functional structures of syntax and semantics are identical (except for the special case).

Forward Composition: X/Y + Y/Z → X/Z

X/Y + Y/Z	–"call" protraction→	(which yields subgoal)
X/Y + Y	–forwards application→	
X	–"return" from protraction→	(which yields result)
X/Z		□

Generalised Forward Composition: X/Y + (Y/W)/Z → (X/W)/Z

X/Y + (Y/W)/Z	–"call" protraction→	(which yields subgoal)
X/Y + Y/W	–"call" protraction→	(which yields subgoal)
X/Y + Y	–forwards application→	
X	–"return" from protraction→	(which yields result)
X/W	–"return" from protraction→	(which yields result)
(X/W)/Z		□

The other common composition rule, Backward Crossed Composition, has not arisen in the George grammar, presumably because of its limited coverage. It is not subsumed by the existing protraction operation, which is only defined for combinations where the second combinand is a forward function. Given the Backward Crossed Composition rule, Y/Z + X\Y → X/Z, it is not hard to imagine an equivalent protraction on the first argument, with Z as the protracted symbol. If such a rule is necessary in a wider coverage version of George, the proof of coverage would be as follows, given a suitable operation *protraction'*. The addition of such an operation need not cause problems of overgeneration, because, like all of the unconventional transformations in George parsing, it is tightly constrained by the symbols in the data – if the operation is necessary in this grammar, then it will apply, by definition, only to the data that make it so.

Backward Crossed Composition: Y/Z + X\Y → X/Z

Y/Z + X\Y	–"call" protraction'→	(which yields subgoal)
Y + X\Y	–backwards application→	
X	–"return" from protraction'→	(which yields result)
X/Z		□

4.3.4. An Alternative Statement of Protraction

Also, one might assume that application of the protraction rules (see 75) are equivalent to the rule in 82, which uses a λ-substitution to swap the arguments to the function around, producing the same behaviour as protraction. Symbols are as before; square brackets are used merely to aid readability.

$$X : \alpha + (Z\backslash X)/Y : \Phi \quad \rightarrow \quad Z/Y : (\lambda x.\lambda y.[[\Phi(y)](x)])(\alpha) \qquad 82$$

This single rule, though, is not general enough for our purposes in George, because it requires a rule for each arity of predicate appearing in position Φ of the above example. To deal with di-transitive verbs we would therefore need a corresponding rules with three new λ's on the right hand side; for tri-transitives, one with four, and so on. We can represent this with a schema, thus, where **Y** represents n forward combinations:

$$X : \alpha + (Z\backslash X)/\mathbf{Y} : \Phi \quad \rightarrow \quad Z/\mathbf{Y} : (\lambda x_1 .... \lambda x_n .[\Phi(x_2,...x_n,x_1)])(\alpha) \qquad 83$$

I suggest therefore that, though the above schema captures the transformation we need, on grounds of elegance alone, the single, universal pair of rules defining protraction are preferable to the choice between a language-dependent number of rules like 82 and the need to introduce meta-rules like 83 into the George Parser.

Perhaps more importantly, the protraction operation captures nicely the intuition behind what we are doing here – the idea already mentioned of holding part of the protracted expression constant while looking ahead to a time when that part will be fully elaborated. This intuition is not captured at all by the rule(s) suggested in 83.

4.3.5. The Procedure of Protraction

Finally, it is worth re-emphasising that, in procedural (implementational) terms, protraction is a process in three parts: the temporary removal and storage of an abstraction operator and its bound symbol; a recursive call of combination rules (possibly including protraction again); and the attachment of the stored abstraction operator and bound symbol in front of the result of the recursive call to give the final result. It is important to understand that this process happens <u>during</u> combination of a <u>single</u> <u>word</u> into an existing partial sentence; the recursive call never involves reading input data.

5. More complicated parsing - Coercions

5.1. Introduction

The foregoing description covers the two lowest levels of parsing with the George Parser. While this is largely adequate for parsing the kind of sentences we need to use to exemplify underspecified reference, some thought has been given to approaches to dealing with harder linguistic problems. Conveniently, it turns out that the same mechanism can be used for doing this as for resolving lexical ambiguity which has been implicitly maintained in adaptable translations. This mechanism bears a strong similarity with the notion of *coercion* in strongly typed programming languages (see, for example, [Brailsford & Walker, 1979]); therefore, I will refer to it by the same term.

A coercion is a transformation performed upon the self-contained translation of an initial sub-sentence in order to enable it to combine with a lexical translation in a way different from that specified by the basic grammar rules and its existing category. In the context of phrase structure grammar, coercions might be viewed as unary grammar rules. The particular transformation performed by any coercion is defined by the syntax, and sometimes by the semantics, of the translation to be coerced; it may or may not also result in changes in the syntactic and/or semantic parts of the translation.

Note that under this definition, type-raising is not a coercion, because it is defined on lexical entries by a lexical redundancy rule. It is therefore not demand-driven, and can (and does) introduce unnecessary non-determinism.

In order to add coercions to the George Parser, we need another general principle, which we can express by means of a new recursive parsing rule. These rules seem to be proliferating – this, though, is the last; and anyway, such proliferation is preferable to a less uniform system where such ideas as coercion or protraction are "hard-wired" into the parser. Specification of such concepts as part of a grammar is preferable, because of the considerably greater flexibility this offers the grammar writer. The new rule specifies that two partial translations may combine to give a (self-contained) result if there exists a coercion ($\rightsquigarrow$) which adapts the first combinand to a partial translation which combines with the second combinand to give that result. The formal statement of the rule is thus:

$$X:\xi + Y:\psi \rightarrow Z:\zeta \quad \Leftarrow \quad X:\xi \rightsquigarrow X':\xi' \quad \wedge \quad X':\xi' + Y:\psi \rightarrow Z:\zeta \qquad 83$$

The $\rightsquigarrow$ symbol denotes the "coerces to" relation, defined by the grammar-writer, and discussed in Section 5.2. Note that, although the rule's recursive nature allows more than one coercion to be applied, coercions are only applicable to the partial sentence (*ie* the left hand side) in a combination. This is because application of coercions to (protractions of) lexical entries is equivalent to the existence of deductive lexical entries or lexical redundancy rules, which we already have. The point of a coercion is to allow adaptability within a partial translation, not between lexical entries.

5.2. Examples of Coercion

A simple example of coercions is that which deals with one form of elliptical noun phrase, the kind where the head noun is omitted in a phrase selecting one of two referents (by colour in this case), as in 84a (where the gap is indicated by ∅). Note that this form of ellipsis is not universally applicable, as in the ill-formed 84b. As an *ad hoc* approximation for the purposes of example, I will suppose that only colours allow this form – though this is clearly not general.

"John owned blue and red t-shirts. The blue ∅ suited him." 84a

* "John owned striped and spotted t-shirts. The striped ∅ suited him." 84b

In the second sentence of 84a, we can transform the n/n category we have at the end of the first noun phrase to an n category, using the coercion defined in 85. This coercion is unusually simple, because of the (non-ideal) non-uniform representation of adjectives in George, which will be covered later. The coercion states that any left-complete phrase lacking a head noun, which would, were it present, become part of a reference whose defining properties are currently only colour (the calls to the *sort_of* and *subsumes* predicates), may be viewed as the completed phrase. As before, upper case letters denote variables and _ is the "don't care" unnamed variable. Lower case Roman letters denote constants; greek letters denote semantic expressions. The *sort_of* relation relates a reference and an expression to the sort of the reference as defined by that expression. (For example, *sort_of*(ref1, [eats,ref1 ˜man!], man) .)

$$C/n(\_) : r\#\alpha \rightsquigarrow C : \alpha \quad \Leftarrow \quad sort\_of(r, \alpha, \sigma) \ \wedge \ \text{colour}\ subsumes\ \sigma \qquad 85$$

Remember that the formula on the right hand side of the ⇐ may in principle be arbitrarily complicated, so a more realistic condition for the applicability of this coercion may easily be imagined here.

The parse of the second sentence of 84a, then, runs as follows (the syntactic category features are omitted for clarity). Recall that the inclusion of a composition step is specific to adjectives and that this is a spurious exception of which details are given in Section 6.

	np/n : ref1#ref1!	+	n/n : blue	
–composition→	np/n : ref1#ref1˜blue!			86a

	np/n : ref1#ref1˜blue!	+	s\np/np : λvar1.λvar2.[suited,var1,var2]	
–"call" protraction→	np/n : ref1#ref1˜blue!	+	s\np : λvar2.[suited,var1,var2]	86b

In satisying the right hand side conditions of the protraction rule, 75, we make (recursive) use of the combination rules, in an environment in which var1 is a constant. The attempt to match this new combination with the left hand side of a combination rule will fail in all cases except with the rule which introduces coercion of the left hand combinand. Syntactically and semantically, this unifies with the coercion given in 85.

	np/n : ref1#ref1˜blue!	+	s\np : λvar2.[suited,var1,var2]	
–coercion→	np : ref1˜blue!	+	s\np : λvar2.[suited,var1,var2]	
–application→	s : [suited,var1,ref1˜blue!]			86c

Now that we have completed this combination to yield a single translation, we are free to return from the recursive call in the protraction rule, which will result in the reinstatement of the λ in front of the protracted expression, after which rightward application may proceed normally.

	s/np : λvar1.[suited,var1,ref1˜blue!]	+	np : ref2˜male!	
–application→	s : [suited,ref2˜male!,ref1˜blue!]			86d

This is the required translation, according to the definitions in Chapter 4.

As a further, more complicated example, we can use coercions to deal with relative clauses which are not explicitly marked with a relative pronoun, as in example 87.

"The man I saw ∅ kicked the donkey." 87

Now, this particular form can only appear when the trace is not the subject of the relative clause – otherwise, the relative pronoun is obligatory. This is exactly the kind of thing coercions are good at expressing – the unification constraints on their introduction may be specified as tightly or as loosely as is necessary (*ie* by means of an arbitrary logical expression), even, should we wish to, down to specifying a coercion applicable only to a

particular partial sentence. Going this far, though, would devalue the George parser and the particular grammar in question as a general solution to the parsing problem.

So what we need to express in this case is that a noun phrase ("The man") with a semantic translation Σ, say, may be coerced into a categorial syntactic function taking object-less transitive verb phrases ("I saw $\varnothing$") and yielding post-modified noun phrases, and that Σ is to be transformed in a corresponding way, if and only if the syntactic transformation is possible.

It so happens that this raises another problem as well – one which is fundamental to incremental parsing. I mentioned in Chapter 3 that post-modification is a linguistic phenomenon which is at odds with the incremental approach. Ideally, if we want to maintain isomorphism between syntactic and semantic structures, we would like to view noun phrase post-modification as the backward application of a categorial function mapping noun phrases to noun phrases. For a post-modified noun phrase at the front of a sentence, this is trivially adequate. However, when we are parsing the noun phrases later in the sentence we have as our translation so far, not, in general, something of category np, but something of category s or s/X where X is some other arbitrary category. For example, suppose we have the following naïve lexical entry for a relative pronoun, where Φ is a semantic function:

$$\text{who} \quad := \quad ((\text{np}/(\text{s}\backslash\text{np}/\text{np}))/\text{np})\backslash\text{np} : \Phi \qquad 88$$

(The lexical entry represents a function applying backwards to a noun phrase to yield a function applying forwards to a noun phrase and then a transitive verb.) Suppose that we are translating the sentence

"The donkey kicks the man who beats him." 89

and that we have reached the stage where we have read, translated and composed the partial sentence "The donkey kicks the man", and have read the "who". Since we have characterised the relative as a function applying backwards to noun phrases, we now need to perform the combination (where α is a semantic expression representing the translation so far):

$$* \qquad \text{s} : \alpha + ((\text{np}/(\text{s}\backslash\text{np}/\text{np}))/\text{np})\backslash\text{np} : \Phi \quad \xrightarrow{\text{application}} \quad ((\text{s}/(\text{s}\backslash\text{np}/\text{np}))/\text{np}) : \Psi \qquad 90$$

which is impossible under our definition of application, because s does not unify with $((\text{np}/(\text{s}\backslash\text{np}/\text{np}))/\text{np})\backslash\text{np}$. Further, Ψ is some expression which we cannot specify – it is impossible directly to pick out the part of the semantic translation which relates to the

noun phrase, because there is no syntactic correspondence with which to unpack the translation. [Pareschi, 1988] suggests a solution to this, using a chart to allow unpicking of the path of compositions leading to the current translation. This solution is unsatisfying here, because it contradicts the initial premise of incremental parsing. Since this postmodification problem is not fundamental to George's area of interest it is enough here to circumvent it; the particular circumvention used here, though, is a good example of the power of adaptable representations, and of the use of coercions to manipulate them. The argument behind my solution runs as follows.

One obvious way of treating this special post-modifying case might be to express it as an exception to the basic grammar with the following coercion, from np to post-modified np (characterised as function from post-modifiers – *nmod*s – to noun phrases):

$$np : \alpha \quad \rightsquigarrow \quad np/nmod : \lambda varN.[varN,\alpha] \blacktriangleright \alpha \qquad 91$$

According to our combination rules, this would be applicable to complete noun phrases just before they combined with the immediately subsequent word if that word had result category nmod. However, this is inadequate, for the same reason as the backwards application approach outlined above: in the case of a noun phrase not at the start of a sentence, the np will have been subsumed into a "higher" category (probably s) by the time it is complete; therefore, the coercion will not unify.

We can, however, adapt this idea, to produce a slightly less elegant form, which seems to do the job. Consider the coercion in 92:

$$X/n : \rho\#\alpha \quad \rightsquigarrow \quad X/nmod/n : \rho\#\lambda varN.[varN,\rho] \blacktriangleright \alpha \qquad 92$$

This coercion will now apply as soon as we attempt to combine any determiner or quantifier (np/n) with the rest of its noun phrase. However, while this is certainly a viable solution, there is a danger that it might introduce a kind of spurious non-determinism, because of the variable on the left hand side of the specification. This is unsatisfying, because the whole point of coercions is to enable adaptability – which is supposed to reduce non-determinism.

So let us consider another solution. An approach using the lexical redundancy rule partly equivalent to the coercion in 92 includes np/n entries as data in the lexicon, and deduces from them (via that lexical rule) a second lexical entry (of category np/nmod/n) for each word of the simpler category. This again solves the problem, but, as a side effect, increases non-determinism in the parse just as much as type raising, which is undesirable.

Now, suppose instead that we were strictly to follow our declared intention in the design of the George system to make translations adaptable and represent ambiguity implicitly. Then, we could use a representation of noun phrases which is syntactically and semantically ambiguous between post-modified and plain noun phrases. We could use a coercion in a way much closer to coercions' supposed purpose: to resolve or at least reduce the ambiguity once we know the correct way to do so.

The effect we want to achieve is this. From one ambiguous translation of category X/n for some X (*ie* a translation which has just incorporated a determiner), we must produce either of the translations shown in 93, using one precisely defined coercion.

$$\begin{aligned} &X/n : refN_1 \# \rho \\ &X/nmod/n : refN_1 \# \lambda varN_2.[varN_2, refN_1] \blacktriangleright \rho \end{aligned} \qquad 93$$

Now, it so happens that the functional expression in GRL for a phrase which is post-modifiable contains all the information contained in its non-post-modifiable equivalent (compare the two translations above). Therefore, if we can supply a coercion which will remove the extra information on demand, we have exactly such an ambiguous notation, ready made, in the form of the post-modifable version above. To introduce such a form into the system is easy: each determiner or quantifier is <u>replaced</u> in the lexicon (NB – lexical non-determinism is <u>not</u> increased) by its post-modifiable form.

Because of the quantifier expansion operation defined in Chapter 4, which ensures that context extensions are always at the fronts of GRL expressions, we can define a very simple coercion to do the simplification we need (something looking for an nmod into something not). The coercion applies after combination of the head noun into the phrase, so the final /n of example 93 has already been combined away. υ is a variable over variable symbols and X a variable over categories. **Q** is an arbitrarily complex series of nested quantifiers.

$$X/nmod : \lambda\upsilon.\ \mathbf{Q}.\ [\upsilon, \alpha] \blacktriangleright \alpha \quad \rightsquigarrow \quad X : \mathbf{Q}.\ \alpha \qquad 94$$

This coercion, unlike the earlier suggestion, cannot repeatedly add categories to the current translation: either it removes the nmod or it does not. Therefore, non-determinism is only introduced if there is explicit justification for it (*eg* if the next word can <u>both</u> introduce an nmod <u>and</u> combine directly with the category X above).

In proposing this mechanism, we seem to have dealt with a serious general problem of incremental parsing, and reduced non-determinism in George at the same time – the

latter because the "non-modified determiner" reading is only ever considered when the coercion rule matches with something following the noun phrase which is not a post-modifier but which can combine with a "plain" noun phrase.

Returning, then, to our particular example,

("The man I saw ∅ kicked the donkey." 87)

we now have a uniform means of incorporating both plain noun phrases and post-modified ones into our translation when they appear in the input.

What we need next is a means of connecting the subject noun phrase of the subordinate clause (category np), "I" in this example, with an item of category nmod in order that we may start to combine the np postmodifier with its antecedent. We need such a connection because of the absence of information from this kind of relative clause which would normally arise from the pronoun – *ie* that explicit relative pronouns yield category nmod. The coercion specified in 95 allows us to make the connection (*r* is a variable ranging over references; *v* ranges over variable symbols). The coercion allows the transformation of a phrase lacking an nmod into one lacking a relative-pronoun-less relative clause; it is introduced by the attempt to combine an object of category X/nmod with one of category X (or of any category which can be reduced to X by λ-protraction), because the left hand side of the general coercion application rule matches the combination.

$$X/\,\mathrm{nmod} : \lambda v.\ \mathbf{Q}.\,[v,r] \blacktriangleright r \rightsquigarrow (X/(s\backslash np/np))/np : \lambda v.\lambda \mathrm{varN}.\ \mathbf{Q}.\,[[\mathrm{varN},r],v] \qquad 95$$

So what we have here is a rule applicable to one particular category in certain circumstances to produce a new form in order to enable a particular combination. As such, it sounds very like (and as *ad hoc* as) Steedman's type raising. However, this is not actually so.

First, the coercion is part of a more general mechanism; unlike type-raising, it does not exist in isolation as a class of operation. Secondly, it is defined genuinely in the spirit of CG – that is, the rule that introduces it is all the definition it needs – as opposed to the general abstract precept that only some categories may be type-raised. Thirdly, it does not introduce lexical ambiguity. Fourthly, and finally, this coercion enables the same combination as is enabled by the existence of a translation of "which" or "that" in the lexicon. The existence and effect of the coercion in a sense corresponds with that of the relative pronoun in language; it may therefore be justified by the same linguistic argument, and so has equally valid status in George.

The effect of introducing a coercion mechanism in an adaptable system is to shuffle some structural ambiguity into non-determinism over which coercion to apply. This is preferable, first on the grounds that no early decision can generally be made on choosing a correct unambiguous structure (so the representation needs to remain ambiguous and adaptable), and secondly because coercions are demand-driven, so choice points are not generated – the decision is simply delayed until enough information is received to make it the correct choice, in line with the basic idea of adaptability.

In conclusion, I should point out that coercion is orthogonal to protraction, in that the actual transformation expressed by the combination rule only ever applies to the left hand combinand, whereas that expressed by the protraction rule only applies to the right. Therefore, because George performs an exhaustive search over the combinations generated by the rules, it does not matter in which order we protract or coerce, even if the operations are interleaved; eventually, the same result(s) will be achieved, albeit in a different (but, in George, irrelevant) order.

5.3. Coercions used in George

There follows a list of the coercions allowed in the current implementation of the George system. The coercions are classified according to function; the full expansion of the category features has been included for completeness. As with the rules specified before, this is a necessary set; however, it seems very unlikely that it might be sufficient.

One obviously missing set of coercions is that covering the verbs "to be" and "to have". These could clearly make possible the replacement of the large number of lexical entries for the numerous different semantic forms with one for each syntactic form, the appropriate coercion from a general translation to a specific one being selected, through unification with the coercion combination rule, by the attempt to combine with, say, a subsequent adjective, noun phrase, or participle.

As before, upper case letters denote variables, and lower case, constants; Roman letters describe (parts of) syntactic data and Greek letters, semantic data which can be viewed as atomic in this context. _ denotes the unnamed variable, "don't care".

Head noun anaphor (for colours only)

sort_of is a relation mapping a reference, r, and an expression in GRL, α, containing r, to the sort of r, σ, as defined by α.

$$C/n(\_) : r\#\alpha \rightsquigarrow C : \alpha \quad \Leftarrow \quad sort\_of(r, \alpha, \sigma) \wedge \text{colour } subsumes\ \sigma$$

The *subsumes* predicate is as specified in Chapter 4. Recall the earlier note that this coercion is an *ad hoc* approximation; the real conditions for application of this coercion are much more complex. Nevertheless, they can be expressed in the arbitrary logical formula in the right hand side of this rule.

Post-Modifiable NP conversion

$$X/nmod(\_,\_) : \lambda v.\ \mathbf{Q}.\ [v,r] \blacktriangleright \alpha \rightsquigarrow X : \mathbf{Q}.\ \alpha$$

Q denotes an arbitrarily complex nested set of quantifiers and abstractions; *r* ranges over references, *v* over variable symbols.

Post-Modification by Relative Clauses without Relative Pronouns

$$X/nmod(N,\_) : \lambda v.\ \mathbf{Q}.\ [v,r] \blacktriangleright \alpha \rightsquigarrow$$
$$(X/(s\backslash np(M,nom)/np(N,acc)))/np(M,nom) : \lambda v.\lambda varP.\ \mathbf{Q}.\ [[varP,r],v] \blacktriangleright \alpha$$

N and M are variables relating the respective numbers of the parts of the references together under unification.

Note that in all these cases, the effect of the coercion may be viewed as mapping from an ambiguous representation to a more specific one – even the coercion for head noun anaphors (above) can be viewed as changing an ambiguous category np/n to the more specific category, n.

6. An Element of Non-Uniformity

Now, unfortunately, there is one element of non-uniformity in these definitions: adjectives (functional category n/n) are not defined semantically as abstractions. This violates the principle assumed here (to make the λ-protraction operation general) that the number of abstractions in the semantics of a word be the same as the arity of its category, and therefore must be treated as a special case. The problem is due to a specific design decision

in George, which was made in order to facilitate the notation and incremental composition of noun phrases: adjectives are treated as composing with determiners and quantifiers, even though they do not have a functional semantic structure. It would disappear if, for example, the George representation were adapted to one where nouns and adjectives were considered as functions over determiners (as is becoming a prevalent trend among CG users) rather than the other way round (which is the *status quo*). However, since this imbalance does not adversely affect the central purpose of the George system (*viz* the analysis of noun phrase reference), it is not necessary to to correct it here.

We do, however, need an extra, regrettably *ad hoc* rule to deal with the case ($\circ$ denotes a special semantic composition for adjectives, defined in Chapter 4, Section 7.1.4):

$$X/n : \Phi + n/n : \alpha \quad \rightarrow \quad X/n : \Phi \circ \alpha \qquad 96$$

The consoling point about this is, of course, that the very irregularity of this rule adds strength to the suggestion above that the problem is indeed derived from violation of the principle of congruence between syntactic and semantic translations.

Note, incidentally, that instead of including a completely spurious composition rule, we could have more uniformly written a second clause for the #-protraction rule to cover this case. However, use of the composition rule indicates more clearly that this is genuinely a special case, which is preferable.

7. Categorial Grammar and Adaptability

Now, I must return to the issue deferred from Section 2.3 of this chapter, regarding the suitability of categorial grammar formalisms for George's kind of parsing. Why is categorial grammar (or at least an equivalent formalism) so well suited to parsing with adaptable representations?

One feature of CG is its ability to express the syntactic form of any well-formed partial sentence, rather than just that of constituents as in, say, Phrase Structure grammars. As mentioned before, this latter function in itself is fundamental to strictly incremental parsing. It is also fundamental to adaptability. To maintain full adaptability of representation, we wish to define coercions, which are transformations between (partial) translations. The specification of such a transformation would be much harder than it is in George if one could not always be sure of the existence of a single, self-contained translation of the sentence being parsed.

Another important feature of categorial grammar in George is the explicit representation of syntactic category in correspondence with the structure of the related semantic information. This is important to adaptability at a practical level: in general, we need all the information we can get if we are successfully to specify the preconditions of coercions. In particular, it would not be helpful if the syntax were (*eg*) encoded in a tree, separate from the semantic representation. The specification of coercions if either of the above requirements were not met would be much less elegant than it is.

On these bases, categorial grammar is ideally suited for the work presented here, and for incremental parsing with adaptable representations in general.

8. Summary

8.1. The George Parser

In this chapter, I have explained the following points about the George Parser.

1. The George process model is a loop, adding translations of new input, strictly word by word, into each one of a set of possible translations-so-far. Failure of a possible translation to combine with a new word causes the translation to be deleted. If a word can combine in more than one way, an appropriate number of new translations replace the old one. If the set of translations becomes empty, the parse has failed.

2. Within the possible translations, the discourse memory of previously parsed complete utterances is maintained separately from the partial translation of the current utterance in each.

3. On receipt of an end of sentence marker, successful utterance completion is detected by checking for certain distinguished syntactic categories, and causes transfer of the current translation and associated reference information to the discourse memory and its reference information.

4. The Parser has no knowledge of the decisions made by the rest of the George system, except where the reference analysis for a translation has failed, in which case the Parser may be instructed to give up on that translation.

5. Sets of references are extracted from the new sentences by the Parser and passed to the DeReferencer.

6. The output of the Parser is formulated in the George Representation Language which is specially designed for the representation of referential expressions and the application of operations fundamental to their analysis by the DeReferencer.

7. The strictly incremental, stack-free nature of the Parser enforces incremental parsing. This provides a strict framework for experiment. It also means that we have at all times a single self-contained (partial) representation of the current utterance, which is a precondition for elegant adaptability.

8. In accordance with the "ambiguity" hypothesis of Chapter 3, Section 2.3, the Parser maintains lexical ambiguity in translations until it is resolved in the input. Coercions help make this possible.

9. Coercions are necessary for parsing adaptably. Incremental parsing requires a self-contained representation of partial sentences. Therefore, CG is an ideal tool.

10. Improvements are made in determinism over existing approaches by systems including either protraction or coercion.

11. Protraction replaces type raising and functional composition, and is demand driven.

12. Coercion allows adaptable representations of multiple readings in one GRL form to be transformed into another representing fewer readings. Non-determinism is reduced because disambiguation is demand driven.

8.2. Parsing Rules

The combination rules for George are as follows. Note that the third basic combination rule, for adjectives, is a special case necessary for this particular semantic representation. The applicability of the combinations is determined by the following factors: unification of the combinands with the left hand side of the rule; successful application of any conditions applied to the rule (including the recursive calls in the protraction and coercion rules); successful evaluation of the resulting expression (including well-typed-ness).

Basic Combination

First word:	$\varnothing : \varnothing + X : \Phi$	$\rightarrow$	$X : \Phi$	
Forwards Application:	$X/Y : \Phi + Y : \alpha$	$\rightarrow$	$X : \Phi(\alpha)$	
Backwards Application:	$Y : \alpha + X\backslash Y : \Phi$	$\rightarrow$	$X : \Phi(\alpha)$	
Composition for Adjectives:	$X/n : \Phi + n/n : \alpha$	$\rightarrow$	$X/n : \Phi \circ \alpha$	
End of Sentence:	$X : \Phi + \varnothing : \varnothing$	$\rightarrow$	$X : \Phi$	$\Leftarrow$ *distinguished*(X)

Combination with Protraction

$X : \Phi + V/U : \lambda v.\Psi \rightarrow Y/U : \lambda v.\Xi \Leftarrow X : \Phi + V : \Psi \rightarrow Y : \Xi$

$X : \Phi + V/U : r\#\Psi \rightarrow Y/U : r\#\Xi \Leftarrow X : \Phi + V : \Psi \rightarrow Y : \Xi$

Combination with Coercion

$X : \xi + Y : \psi \rightarrow Z : \zeta \Leftarrow X : \xi \rightsquigarrow X' : \xi' \wedge X' : \xi' + Y : \psi \rightarrow Z : \zeta$

8.3. Coercions

Head noun anaphor (for colours only)

$C/n(\_) : r\#\alpha \rightsquigarrow C : \alpha \Leftarrow sort\_of(r, \alpha, \sigma) \wedge$ colour *subsumes* σ

Post-Modifiable NP conversion

$X/nmod(\_,\_) : \lambda v.\ \mathbf{Q}.\ [v,r] \triangleright \alpha \rightsquigarrow X : \mathbf{Q}.\ \alpha$

Post-Modification by Relative Clauses without Relative Pronouns

$X/nmod(N,\_) : \lambda v.\ \mathbf{Q}.\ [v,r] \triangleright \alpha \rightsquigarrow$
$(X/(s\backslash np(M,nom)/np(N,acc)))/np(M,nom) : \lambda v.\lambda varP.\ \mathbf{Q}.\ [[varP,r],v] \triangleright \alpha$

9. Afterword

This chapter has discussed the operation of the George Parser in its full detail. This in itself emphasises how simple and uniform a system George is. While there is a fair amount of work which could still be done to follow up the possibilities of coercion driven incremental parsing, enough has been done to enable an approach to the central work being presented here, and therefore the parser must (regretfully) be left at this stage.

Even in this embryonic form, the George parser has proved to be an efficient and reliable framework within which to develop ideas about incremental reference evaluation. These ideas will be explained in the next two chapters.

Chapter 6
The George DeReferencer

Abstract

A general overview of the process model of the George DeReferencer is presented. The rules which determine the behaviour of the process are introduced.

The rules are discussed in the same step-by-step style as in earlier work by Mellish. Each linguistic possibility is dealt with separately, the relevant parts of the dereferencing mechanism being introduced at the appropriate points.

I first deal with simple cases where each singular reference introduces a new entity and develop this into definite singular reference. Post-modified noun-phrases are dealt with through the idea of *context extension*, and, finally, the notation for and behaviour of plural references is introduced.

In particular, this last is done with only minimal changes to the overall mechanism developed initially. Thus, I conclude that the George DeReferencer is a particularly simple and uniform solution to the basic problems of reference.

This chapter introduces a new treatment of noun-phrase post-modification, in particular by relative clauses, and an associated view of pattern matching in this context which partly vindicates the decision to separate references and entity tokens in George.

1. Introduction

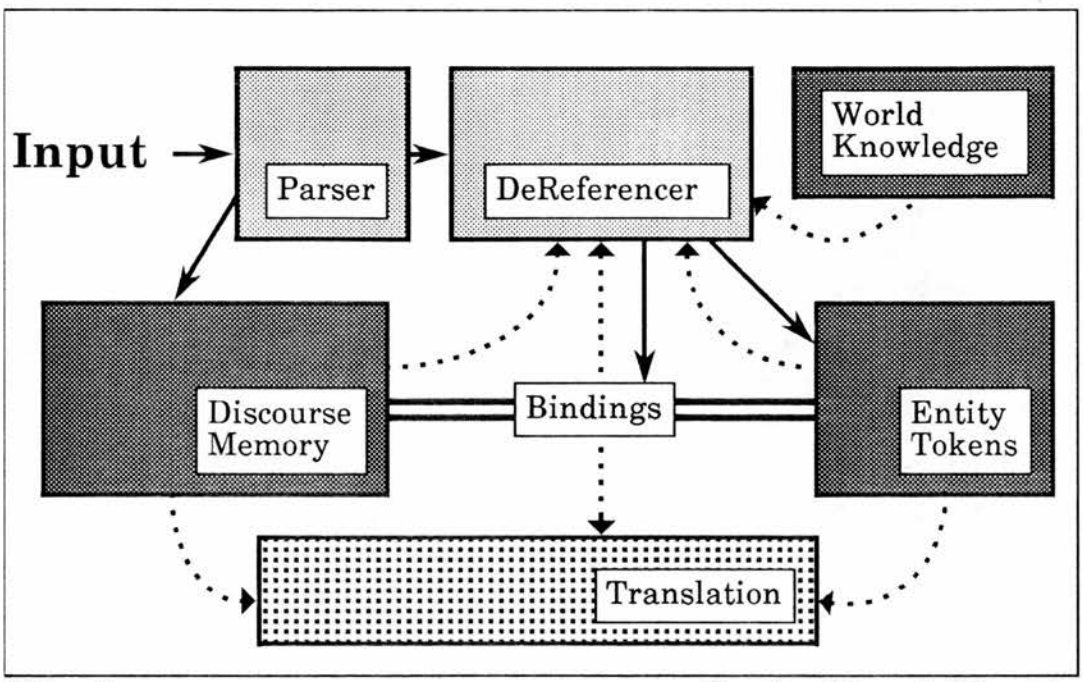
It should be clear by this stage of our discussion that the aspects of this work related to reference evaluation are heavily motivated by earlier advances with comparable ideas by Chris Mellish. A detailed comparison with Mellish's work (and that of Bonnie Webber [Webber, 1979], which is also fundamental to George) will be made in Chapter 8.

This chapter explains the basic behaviour of the George system with respect to reference. As such, it is largely devoted to the reconstruction of the ideas presented in [Mellish, 1981] and [Webber, 1979], in a framework appropriate to the view of language understanding presented in the first three chapters of this document. It is necessary, in particular, to show that the use of adaptable representations need not compromise the ability of a system to perform those reference analysis tasks which Mellish and Webber's systems could perform. Beyond this, the view here is rather different from both of these earlier theses, in that no attempt is made to find one general universal rule for reference analysis. Rather the problem is divided up according to the linguistic properties which define the data, and individual rules are formulated based on these properties.

2. The George DeReferencer

2.1. The Top Level

In Chapter 3, I explained that the George DeReferencer runs in (pseudo-)parallel with the George Parser in each of the basic George systems making up the set of possible translations for the input utterance. For convenience, Figure 2a, the schematic diagram of the layout of the basic George system given before, is reproduced here.



(Figure 2a: Functional Layout of George)

The DeReferencer is a separate process from the Parser, autonomous, except that it is dependent on the Parser for input and that should either process fail (and therefore die) the other will kill itself in sympathy, along with the whole of the rest of the possible translation. Direct input from the Parser to the DeReferencer is in the form of sets of references – that is to say, reference symbols, as defined in Chapter 4, and the sort, number, definiteness and context extension information associated with them by the discourse input – extracted from the translation in GRL of the Parser's input. Either the elements of these sets are new references or they already exist; this is detectable by comparing reference symbols, which are unique to each reference in the discourse. If they already exist, the new version, just received from the Parser, replaces the old version known to the DeReferencer, on the grounds that it may have been adapted (*eg* by the

noun-phrase post-modifier coercion) and that consistency between the different parts of the system must be maintained. The DeReferencer is allowed read access to the Discourse Memory, entity tokens, and bindings (both local to the current sentence and global across the whole discourse) within the current translation. It may also write its own results – in the form of creation, refinement and deletion of individual bindings and/or entity tokens data – into the appropriate database. Finally, the DeReferencer has read access to knowledge of discourse world sorts and axioms for (limited) real world deduction to enable it to reason about reference.

Effectively, George simulates an interleaved or parallel execution of the Parser and DeReferencer, which is certainly what is needed (having forbidden backtracking and given an incremental evaluation of semantics) in order to enforce strictly incremental evaluation of reference. While it would not be particularly fruitful to build such an implementation in existing parallel (logic) programming languages (because the relevant technology is still under development), it is clear, because of the almost trivially simple protocols for inter-process communication in George (*ie* one unidirectional data stream and one bidirectional suicide pact), that genuine parallelisation of the basic George system on this coarse level would not be hard.

2.2. Some Terms

It will be useful here to (re-)explain some terminology and some of the structure of George. Referring to the restated diagram above, recall from Chapter 5 that the *Discourse Memory* is the repository for the "surface form" translations of the input to the Parser.

As mentioned above, the DeReferencer is able to write information into two logically separate databases. One contains information about *bindings* – that is to say, specifications of a (possible) mapping between the content of the Discourse Memory and that of the Entity Tokens Database; the other, namely that same *Entity Tokens* database, contains information hypothesised by the DeReferencer about the *entities* taking part in the discourse.

The *World Knowledge* database supplies knowledge about the property/value pairs which define *sorts* (Sort and SortDef, as defined in Chapter 4). Reasoning about the sorts is performed in terms of consistency and subsumption (again, as defined in Chapter 4), and is encoded in the George algorithm. The limited *discourse world inference* available in George is also encoded in the program, though a genuinely separate database would be preferable for theoretical purity.

Bindings are relations between sets of references and sets of entity tokens. It will be useful to distinguish the case where the set of entity tokens has only one member. I will call this a *singleton binding*.

Finally, the combination of the Discourse Memory, the Entity Tokens and the Bindings databases gives us the finished translation, subject to the limitations of the current George implementation, which does not include the *Discourse Entity* level of reference mentioned in Chapter 3.

2.3. The George DeReferencing Algorithm

The top-level algorithm of the George DeReferencer is as follows. Parts marked ① to ⑥ are lower level sections which will be explained later, ⑥ being covered in Chapter 7. Input is in the form of sets of references (output by the Parser, as explained in Chapter 3) Output consists of creating and deleting bindings and entity tokens and their associated information in the respective databases; and in causing a complete translation to fail if its reference analysis fails.

On receipt of a set of references from the Parser, for each reference symbol in the set, perform each one of the following actions.

If the reference symbol is new to the DeReferencer① then

1. a. create a new binding for it, then...
   b. If its reference is definite②, find the entity token(s) to which it may refer③ and insert them into the binding;
   c. If its reference is indefinite②, create a new entity token and insert it into the binding;

otherwise

2. a. look up its existing binding, then...
   b. Combine any new information④ about the reference with that already contained in the binding, discarding any erstwhile candidate entity tokens which are now made inconsistent③. Check that entity tokens removed are not left newly unbound; if any are, remove them from the Entity Tokens database.

3. If the binding of this reference contains no entity tokens (*ie* if the reference is completely new, or no existing token was found to be consistent, or all the tokens previously found to be consistent were deleted on grounds of inconsistency on this

latest iteration), search backwards through discourse memory for expressions containing references bound to entity tokens of consistent sort with this reference. Apply the underspecified reference mechanism⑥ to each such expression until one is found which yields success under that mechanism; then rewrite the bindings and if necessary the expression itself to create entity tokens for the bindings in the current sentence.

Finally,

4. If the reference is definite and its binding contains no entity tokens (*ie* if all the tokens previously found to be consistent were deleted on grounds of inconsistency on this latest iteration), create a new entity token with the appropriate properties⑤ and insert it into both the Entity Tokens database and the binding.

The aspects of George's operation which extend previous reference analysis work are to be found in sections marked ③ (in particular) and ④, and in part 3. These and the other labelled sections will be explained in detail below, step by step; and a summary is presented at the end of this chapter. Part 3 of the algorithm is the main focus of the work presented here. It will be covered in Chapter 7.

As in the design of the George Parser, an attempt has been made here to divide the operation of the DeReferencing system into a simple, general, repetitive procedure (given above), and a set of more declarative rules which determine its exact action within that procedure. The rules, defined in terms of primitives such as consistency (as in Chapter 4), will be summarised later, when the ideas expressed in them have been developed through example. This division is certainly desirable for the purposes of research, since it enables us to see clearly which effects are caused by procedural interaction (which is important in terms both of examining incremental parsing and reference analysis and of general (pseudo-)parallel processing) and which effects result from our specification of the rules defining the process.

These rules, then, define the operation of the sub-algorithms mentioned above; they will be the focus of the rest of this chapter, since they and the behaviour they describe constitute the central theme of this research.

3. Analysing Noun Phrase Reference

3.1. Introduction

In this and the subsequent sections, I develop the process of reference analysis from the simplest possible case to the most complicated kind of quantified reference covered by George. As did Mellish, I will start with simple examples and elaborate, the idea being that, having solved the most basic case, we can "try to retain the simplicity of the system which works for this case, whilst gradually introducing more and more sophisticated features to cope with the obvious drawbacks" ([Mellish, 1985], p38). In each case, I will summarise the system's behaviour in a rule, which will be a formal restatement of part of the algorithm specified in the last section. It is worth mentioning here that all the GRL expressions and bindings used for the running example in this chapter were taken directly from the output of the working system[1]. The full output is given in Appendix B.

3.2. The Simplest Case

3.2.1. Introduction

First, then, we must consider the very simplest case possible – that where a noun phrase (*ie* in George terms, a reference) just introduces a new entity. This is, of course, to an extent begging part of the original question – since we cannot in general detect when a noun phrase really is introductory. I will in this respect make the common simplifying assumption that an indefinite reference introduces a new entity token, and that indefinite references are therefore never non-introductory.

Note, incidentally, that what we are dealing with here is not the simplest possible kind of reference (for the first sentence of the example contains post-modified noun phrases), but the simplest possible response by the DeReferencing system to the appearance of a reference at the output of the Parser.

The example which Mellish gives at this point in his explanation is shown in 105.

"A particle of mass b rests on a smooth table." 105

1: Some symbols and layout have been changed to improve readability.

This example is specialised to the domain of data Mellish was considering (*viz* A-level mechanics problems), and is inappropriate for my purposes because I have not considered valued modifiers like "of mass b". Instead, I will explain the comparable, but more suitable, example shown in 106.

"A man who is unkind beats a brown donkey." 106

3.2.2. Parsing a Simple Introductory Noun Phrase

So now let us consider the detailed trace of George's operation for sentence 106. As I explained in Chapter 3, the input to the DeReferencer from the Parser is in the form of sets of George references extracted by the Parser's output stage from increasingly (*ie* word by word) detailed approximations to the complete translation of the input utterance.

The first datum received from the Parser arises from input of the word "A". It is a set containing one simple unsorted indefinite reference (as defined in Chapter 4), simply

{ ref1 } 107

Now, we know that this reference is indefinite, because it does not contain the ! operator. This is test ② in the higher level algorithm given before. We also know that it contains a new reference symbol (NB as opposed to referring *to a new discourse entity*, which is not the same thing) simply because the DeReferencer has no previous record of it. This is test ① – a simple search and textual comparison. We conclude, therefore, we must add the reference to the new bindings list. By definition, we now need an entity token to which to bind it; so we create one, which is step ⑤ in the higher level algorithm. So far, we have used Rule 1. (A complete list of the dereferencing rules is given for quick reference in Appendix C.)

A new indefinite simple reference is always bound to a new entity token created for that purpose.

Rule 1: *Simple Indefinite Reference*

In the top-level George algorithm given above, we have now completed a whole pass: for the only reference supplied, we followed part 1a and then part 3. In doing so, we created a

new entity token, say e1, which we must add to our Entity Tokens database, and a corresponding binding. These are shown in 108a and b, respectively.

$$\{e1\} \qquad 108a$$

$$\langle \{ref1\}, \{e1\} \rangle \qquad 108b$$

The general form of these bindings is as follows. The binding is an ordered pair of sets, the first being of references, and the second of entity tokens. The binding may be thought of as connecting the Discourse Memory with the approximations (*viz* entity tokens) George has made to the discourse entities (specified by the entity tokens) in the discourse world. In most of the examples in this chapter, I will omit to state the explicit list of entity tokens, since it is trivially deducible from the bindings and only clutters the explanation.

In most cases, the set of references is singleton. The only possibility (in general) which can give rise to a non-singleton set is that in which two or more references are explicitly coreferential (*eg* are connected by an equative) and marked as such by the application of the *coref* operator in the Discourse Memory. The binding containing the set of references may then be thought of as a conjunctive form, making the single binding equivalent to a collection of bindings between each member of the set of references and the entire set of entity tokens (see below) at the other end of the binding. The chosen notation is preferable to separate bindings because it enables George to maintain consistency between such coreferential references implicitly, which is much easier than if they were split into separate bindings. (An alternative view might be one of unification of references.)

The set of entities at the other end of a binding is, in the same sense as in Mellish's use of the term, a *candidate set*; that is to say, each entity token symbol contained therein is the name of an entity token which is proposed by George as a possible referent for the reference(s) in the binding. Various criteria determine whether such a binding is referentially well-formed in the final analysis: for example, to be well-formed, simple (singular) references must be bound to an entity token which was introduced by a simple reference (so that a singular reference may not refer to a plural object). Note that the candidate set is not always a disjunction, although it may look like one in the first simple examples; indexed (plural) references may be bound to more than one entity token, allowing reference to split, non-uniform sets, in which case the interpretation is conjunctive. I will give examples of these in Section 6.4.

Next, we read the word, "man". This results in a new modified output from the Parser:

$$\{ref1\ \tilde{}man\} \qquad 109$$

Now we apply our algorithm again, this time following option 2a first, because the reference symbol is already known (test ①). We have simply to add to the binding the new sort information (④ in the higher level algorithm) to give:

$$\langle \{ \text{ref1}^{\frown}\text{man} \}, \{ \text{e1} \} \rangle \qquad 110$$

The addition of the sort information to the reference in the binding, rather than to the entity token directly, expresses the fact that the correctness of the binding is conditional upon consistency of that sort information with any that may subsequently arrive from other bindings to the entity token. If the new information is to be considered genuinely applicable to any non-empty subset of the set of entity tokens in the binding, it must first be consistent (in the formal way defined in Chapter 4) with any already applied by this reference – otherwise, the reference is self-contradictory (as in "Colourless green ideas...") and does not make sense in the current discourse world (test ③). Second, (and only in the case of non-introductory reference, for obvious reasons) it must be consistent with any sort and number information calculated and supposed to be correct (more about which later) from previous and subsequent utterances, which is stored in the Entity Tokens database.

In this case, the first of these conditions is trivially fulfilled, since there is no existing sort information in the reference. The second is also trivially true, since this entity token was introduced by this reference – and therefore there can be no information associated with it by previous references with which it might be inconsistent.

Finally, certain criteria (covering effects like non-coreference within clauses, formally stated in Rule 6) are applied to the binding. If they are successful, and if the binding, for a singular reference, is singleton, the sort information passes from the reference part of the binding to the entity token in the Entity Tokens database. In this example, the criteria are successful. The binding can only be singleton, because of the way it was created; and so the sort information passes immediately to the entity token. This was Rule 2.

New sort information applied to a simple reference is passed to the entity token in its candidate set only when that set (and hence the binding) is singleton. Otherwise, any candidate entity tokens which are inconsistent with the new information are removed from the candidate set.

Rule 2: *Sort Refinement*

3.3. Parsing a Relative Clause

The foregoing section, then, covers how we deal with indefinite introductory cases in George. Now we move on to parse the next word in our example, "who". In George, "who" has two lexical entries, one of which covers the usage where "who" is synonymous with "whom" (ie where the relative pronoun does not refer to the subject of the subordinate clause). For the sake of clarity in our example, I will explain only the behaviour of the translation arising from combination with the entry leading to success when parsing this sentence (*viz* that where the gap in the relative clause is in the subject position). The other translation is expressed by a basic George system logically independent of the successful one, and, anyway, dies after input of the next word – as we would wish – because of unification failure between its own syntax and that of the new word. The set of references resulting from the eventually successful combination is:

$$\{[\text{var1},\text{ref1}^{\frown}\text{man}] \blacktriangleright \text{ref1}^{\frown}\text{man}\} \qquad 112$$

At this stage, if the context extension (the predicate to the left of the ▸ operator), were closed (as defined in Chapter 4), the DeReferencer would compare the context extension with previous sentences in the discourse in an attempt to use the information in it to refine references. The detail of this comparison will be covered in Section 5.4. However, since the expression contains the variable, var1, the predicate is not yet closed, and is therefore too general to match meaningfully with a finished translation. Because of this, the DeReferencer simply attempts to refine e1 by incorporating any further information conveyed by the simple sorted reference embedded in the context extension, in the usual way. In this case there is none, and so we have finished this pass of the higher level algorithm.

When we read the word "is" we are presented with the following set of references (again, for purposes of example, ignoring lexical entries leading to an incorrect translation):

$$\{\text{coref}(\text{var2}, \text{ref1}^{\frown}\text{man}) \blacktriangleright \text{ref1}^{\frown}\text{man}\} \qquad 113$$

The DeReferencer's behaviour with this reference is the same as at the previous stage explained above, for the same reasons.

Finally, in this noun phrase, we read the word "unkind", and receive the following reference from the Parser:

{ coref(ref2~unkind, ref1~man) ▸ ref1~man } 114

Now, the appearance of the *coref* operator states that the two references which are its arguments are co-referential. Therefore, we can economically represent the reference by transforming the binding given in 108 to that in 115a and applying the same criteria as before to license transfer of the sort information to the entity token, yielding the entity token set shown in 115b.

⟨{ ref1~man, ref2~unkind }, { e1 }⟩ 115a

{ e1~man~unkind } 115b

As explained before, the pair of references in the binding expresses the coreference of ref1 and ref2; and the singleton set of entity token symbols is the set of candidates for reference by those references.

Now, as it does after reading every new word in the current sentence, the DeReferencer again attempts to compare the context extension in 114 with the information already conveyed by the discourse. (This attempt is included in part ③ of the higher level algorithm.) In this case, the expression will not match with any preceding clause, simply because none exists. The context extension therefore has no referential effect. Following the successful application of the structural criteria mentioned before, the sort information from the two reference symbols is free to be passed to the entity token.

So, we have now completed the first noun phrase. To summarise, we have a single translation of the surface form of the referring expression, linked by an explicit binding to a single database entry containing all the information known about the entity as it exists in the discourse world.

3.4. Parsing a Transitive Verb Phrase

Let us now proceed quickly through the rest of our example sentence. We next read the verb "beats". Within the Parser, this results (through protraction) in the production of a fairly complicated λ-expression; however, the set of references extracted from it by the Parser's output stage is the same as that in 114 above. Therefore, there is no change in the DeReferencer's behaviour. Next, we read "a", producing this set of references:

{ coref(ref2~unkind, ref1~man) ▸ ref1~man, ref3 } 116

and this correspondent set of bindings:

$$\{\langle\{\text{ref1}\tilde{}\text{man}, \text{ref2}\tilde{}\text{unkind}\}, \{\text{e1}\}\rangle, \langle\{\text{ref3}\}, \{\text{e2}\}\rangle\} \quad 117$$

The binding to e1 is unchanged from before; the binding to e2 is produced in exactly the same way as was the initial binding between ref1 and e1. When we go on to read "brown" and then "donkey" we are given by the Parser the sets of references shown in 118a and b respectively, with the new information propagating along the binding, as it were, to arrive at e2 in the Entity Tokens database at each stage. The propagation is licensed by the singleton nature of the binding.

$$\{\text{coref}(\text{ref2}\tilde{}\text{unkind}, \text{ref1}\tilde{}\text{man}) \blacktriangleright \text{ref1}\tilde{}\text{man}, \text{ref3}\tilde{}\text{brown}\} \quad 118a$$

$$\{\text{coref}(\text{ref2}\tilde{}\text{unkind}, \text{ref1}\tilde{}\text{man}) \blacktriangleright \text{ref1}\tilde{}\text{man}, \text{ref3}\tilde{}\text{brown}\tilde{}\text{donkey}\} \quad 118b$$

In the final analysis, then, the output from the program at this stage is as shown in Figure

```
 ** George parser **

State 0:

::::::::::::::::::::::::::::::::::::::::::::::::::::::::

>> A man who is unkind beats a brown donkey.
================================================
End of Sentence

 ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),
          ref3~brown~donkey,
          ref1~man]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey

>>
```

Figure 6: *George output for "A man who is unkind beats a brown donkey."*

6 (except, as before, for minor textual alterations to improve readability). The user input, once George has been started, is simply the text next to the >> prompt, terminated by a full stop. The heading "State 0:" refers to a simple memory mechanism for storing intermediate stages in a parse, for purposes of experiment. 0 is the empty condition.

3.5. Simple Definite Reference Evaluation

We will now progress to the next simplest kind of reference – from George's point of view, the kind which occurs when we know a definite reference to be singular. For the sake of example (for at this point Mellish stops giving individual worked examples), I will continue with our example of the brutalised donkey, with the input sentence in 119. Note that the continuation aspect is important – George deals in discourses, not just in individual utterances.

"A woman feeds the donkey." 119

The behaviour of the system for the first three words is as before (the sentence is designed to make the example useful later as well as being interesting here). After processing "feeds" we are left with the bindings shown in 120 and the entity tokens in 121.

{⟨{ref1~man, ref2~unkind}, {e1}⟩, ⟨{ref3~brown~donkey}, {e2}⟩,
⟨{ref4~woman}, {e3}⟩} 120

{e1~man~unkind, e2~brown~donkey, e3~woman} 121

Next, we input the word "the". This introduces a new reference of a kind new to our discussion of the DeReferencer: a *simple unsorted definite reference*. The reference is shown in 122.

ref5! 122

When we attempt to dereference the set of references including this new one, the old ones are dealt with as before, and there is no change in their bindings, because they are being no further refined (*ie* their information content is not being changed) by the new input. However, when we deal with the new reference, we follow a new path through the top level algorithm; namely, part 1b. In full detail, the reasoning is as follows. We determine that the reference symbol is new (① in the top-level algorithm) from the simple fact that we have no prior record of its reference symbol. We determine that it is definite (②) because it contains the ! operator. Having done so, we need to find which entity tokens it may refer to; and this process, ③, requires a little more explanation.

At this stage, the DeReferencer has very little to go on; it knows nothing about the entity to which the new reference is bound except that it is definite and singular (and in the latter I include syntactically singular generics, the referential properties of which would

be dealt with on the level of token-entity specification in George, by *co-specification* between entity tokens, which will be covered in Chapter 9). Therefore, the application of rules about sort information will not cause refinement of the candidate set, because there is no information yet available with which to make comparisons. The only rules we can apply at this point are as follows. First, we know that the reference is definite – and this has already affected our paths through the higher level algorithm, in choosing part 1b as well as part 1a. Second, we know that the reference is singular (or, formally, *simple*) because it is not indexed. This means (according to Rule 4, below) that it can only be bound to entity tokens which have been introduced by other singular (simple) noun phrases or explicitly derived from existing sets (by the processes explained later in this chapter) – these are the only entity tokens that we can be sure specify individuals.

So, under the weak constraints we have available at the moment, we can produce a binding as shown in 123. Note in particular that all the possible candidates have been identified, under non-contradiction; that is, we have an open world assumption. As no constraining properties have yet been applied, Mellish's and Haddock's (closed world) requirement of explicit consistency would not work here. (Exhaustively selecting elements of a candidate set in this way is not a general solution on its own. The initial selection ultimately needs to be filtered by a focus algorithm, as I explained in Chapter 2.)

$$\langle \{ \text{ref5!} \}, \{ \text{e1}, \text{e2}, \text{e3} \} \rangle \qquad 123$$

Here, we can see explicitly for the first time that (as has been implicitly the case all along) the set of entity tokens in the binding is a *set of candidate entity tokens* and not a set of definitely selected bound tokens. This behaviour is encapsulated formally in Rule 3 which

The initial candidate set of a new simple definite reference always initially contains all the entity tokens in the system which are both sort and number consistent with the reference.

Rule 3: *Simple Definite Reference*

refers to the rule of number consistency, Rule 4.

Next, we add the word "donkey" to our input. This results in the presentation of a newly refined sorted definite reference to the DeReferencer, as shown in 124.

$$\{ \text{ref5}\hat{}\text{donkey!} \} \qquad 124$$

1. **Simple references may not have as candidates entity tokens which are bound one-to-one to any indexed reference;**

2. **A binding involving an indexed reference must contain either**
   a) more than one entity token; or
   b) at least one entity token bound singleton to an indexed reference.

3. **Any two indexed references singleton bound to the same entity token must have equal upper bounds.**

Rule 4: *Number Consistency*

Following paths 2a and 2b in the higher level algorithm, and in particular sub-process ④, we now add this new information to the reference in the binding above, and test for consistency (as defined in Chapter 4) with the sorts of the entity tokens named in the binding (and listed in full detail in the Entity Tokens database). Because of the particular defining properties of the sorts involved here (*man* and *woman*, as opposed to *donkey*), neither e1 nor e3 is consistent with the reference; therefore, they are both removed from the binding. The binding is now one to one, so George assumes that the sort information may pass from the reference to the entity token (which in this case tells us nothing new – e2 is already a donkey), and we are left with the overall translation of the whole discourse so far shown in Figure 7. This behaviour is governed by Rule 2.

Application of this rule is equivalent in principle to [Haddock, 1989]'s network *node consistency*, which is in turn exactly equivalent to part of [Mellish, 1981]'s network consistency algorithm. In practice, the behaviour is different because George uses its open world notion of consistency (*viz* non-contradiction) for selecting candidates, rather than Mellish's and Haddock's stricter closed world, explicit consistency requirement.

3.6. Summary of Simple (Singular) Reference in George

3.6.1. Introduction

Let us now pause for a moment and summarise the process covered so far. We have looked at the behaviour of the George DeReferencer in two very simple cases. One of these, invoked by a *simple indefinite reference*, is viewed purely as an introductory reference.

```
>> A woman feeds the donkey.
================================================
End of Sentence

 ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),
            ref3~brown~donkey,
            ref1~man]
 ** s:[feed(pres,pres,perf,act,indic),ref5!~donkey,ref4~woman]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey

>>
```

Figure 7: *Continued George output for "A woman feeds the donkey."*

The other, introduced by a *simple definite reference*, is regarded as referential on the assumption that it has at all times during processing of the discourse a non-empty *candidate set* of entity tokens to which it may be bound.

3.6.2. Simple Indefinite Reference

When a reference is encountered which does not contain the definite operator, it is assumed to be introductory, so no attempt is made to find entity tokens introduced earlier in the discourse to which it might refer. This results in the creation of an initially empty binding for the indefinite reference, which is immediately filled by the creation of a new entity token. Subsequently, new information leads to refinement of reference as detailed below, with the advantage that this is a case where the binding is known by definition to be singleton from the start.

The case where an indefinite reference is not (in the usual sense) introductory but, rather, generic is dealt with by an operation not detailed here, called *cospecification*. Cospecification is a relation between sets of entity tokens, and so treatment of indefinite singular generics is covered by the introductory view explained above; the rest of the work is done at the third layer of reference evaluation mentioned before, which would attempt to determine from the semantics and pragmatics of the introductory sentence whether or not cospecification was the correct operation to use. Cospecification will be covered in more detail in Chapter 9.

3.6.3. Simple Definite Reference

When a reference is encountered which does contain the definite operator, it is assumed in the first instance to be referential. All the entity tokens with which it is *referentially consistent* (see below) are collected and their names are placed in its binding.

If the reference is context extended and the context extension is closed (as defined in Chapter 4), then the context extension predicate(s) is(are) compared with closed predicates in the preceding discourse, in a way which will be explained in detail later. If this comparison is unsuccessful, George hypothesises that the post-modifier which gave rise to the extended reference is non-restrictive (more about which shortly). Thenceforward, the information about the reference, both in its context extension and in the main body of the reference, is refined in the same way as for any other reference.

The case where a definite reference is generic is rather harder to deal with than that of indefinite generics. If it so happens that there is no entity token already existing to which the reference may be bound, then the creation of a new one (see below) places us in the same position as with the indefinite generic above. However, if this is not the case, we must use similar measures at the entity token level as are suggested above at the level of (co)specification, which, as mentioned before, is outside the scope of this discussion.

Finally, if the candidate set for a simple definite reference is empty, the reference is assumed to be introductory and an entity token is created as in the indefinite case.

3.6.4. Referential Consistency

The overall concept of referential consistency at the binding level is divided into two parts. First is the notion of consistency of sort defined in Chapter 4 under an open world assumption; this needs no further coverage here. Second is consistency of cardinality (Rule 4), which I will demonstrate further as I introduce more complex reference in our running example. For the present, it is adequate to assume that all references are consistent in cardinality, because, at this stage of our discussion, we can only introduce singular references and entity tokens (again ignoring the separate issue of generics).

3.6.5. Refining Reference

Each time the DeReferencer receives an updated set of references from the Parser, a change may have been made in the information contained therein. If a change has been made (*ie* a word has been encountered which contributes to the information contained in a reference), it will apply only to one reference, and therefore to one binding, because of the incremental nature of the parsing process. If the new input made no such contribution there will be no change (*eg* if the input was a verb – see Chapter 9 for a discussion of how selectional constraints on verbs might be incorporated into George).

Initially, any new information is added to the reference "end" of the relevant binding. Immediately, any entity tokens whose sorts are inconsistent with the new information are removed from the entity "end". If the binding is subsequently contains only one entity token, it is assumed to be correct, and all the information in the binding is passed directly to the entity token itself in the Entity Tokens database. This may or may not result in the addition of new information to the entity token – the information from the reference may be subsumed by that already present.

3.6.6. Structural Criteria

So far, we have seen one structural criterion which may apply following the addition of new reference information. We might call this the "closed non-co-reference rule"; it will be formally specified as Rule 5, in Section 4.3. It simply states that no reference may be bound to the same entity token as another appearing in the same closed predicate (*ie* the same clause), unless the two have the same reference symbol. (Note that the coref operator is not an exception to this – a coref expression is not formally a closed predicate, because the coref operator is syncategorematic in GRL and not a member of Pred.) The rule makes explicit the intuition that the references in, for example, "He kicked him." are not normally coreferential, while those in "He kicked himself." definitely are. As it happens, this rule does not affect the reference in our example so far; however, it is useful to introduce it at this stage, so that the reader is aware of the existence and nature of these integrity-constraint-like rules. It will be explained in more detail when it actually applies, in the next example. Representation of reflexives will be discussed, as potential further work, in Chapter 9.

It is useful to express this rule separately from the other rules being applied here (*eg* sort refinement), because it is different in that we need to apply it to every reference in the discourse every time new information about any reference is introduced. The reason for this will become clearer during the next example; suffice it to say for now that this rule and its repeated application allows us to disambiguate some references which are not disambiguable by sort or number information without resorting to harder concepts like focus. The repeated application is justified in particular because some such ambiguity is not resolved within the sentence in which it was introduced, but in subsequent sentences. The application might be viewed as comparable with the enforcement of integrity constraints in relational and deductive databases.

4. Context-Extended Simple Reference

4.1. Context Extension and Noun Phrase Post-Modification

Before we move on from the simple definite case, it will be useful to discuss another possibility in the current simple context. In the initial sentence of this running example, we saw George represent and analyse the reference of a relative clause without appeal to the property of restrictiveness; in any useful language system, we will wish to account for both non-restrictive and restrictive uses of noun-phrase modifiers. George deals with the issue of the difference between these two kinds of relative (and, for that matter, the corresponding adjectival and prepositional usage, too) in a new and very simple way.

Any relative clause translated into GRL is inserted into the Discourse Memory along with the main clause to which it is subordinate; it is explicitly recognisable as subordinate from its position to the left of a context extension operator, ▸. However, in George, no further distinction is made regarding the syntactic or semantic attachment of the modifier – it is viewed as modifying the reference and not the noun or noun phrase (for restrictive and non-restrictive, respectively), as some linguists might prefer. The reasons for this will become clear below.

Before discussing those reasons, let us define the two different kinds of relative. First, a restrictive relative is one which is intended to reduce the candidate set (or its intension) of the reference of which it is part. Conversely, a non-restrictive relative is one which is intended to contribute information to the description of the object(s) referred to by the reference embedded in it. Note that this is not the same as saying that non-restrictives and restrictives respectively do and do not impart new information about a referent. The

addition to a referring expression of information which is inconsistent with some of a candidate set of referents behaves restrictively in exactly the same sense as the addition of information which is inconsistent with the same candidates but which was already known about the others.

Now, in any discourse, we would normally expect a distinction to be made between restrictive and non-restrictive relatives, either by intonation in spoken English or punctuation in written. I suggest that, as long as this syntactic/pragmatic information is felicitous – *ie* as long as it agrees with the practicalities detailed below – this information is effectively irrelevant, and serves only as a secondary check on the semantic well-formed-ness of the discourse. I justify this in the following way, in abstract terms. A worked example is given in the next section.

This view of relative clause (non-)restriction is derived primarily from the practicalities of language analysis. In a sense, the actual (as opposed to intended) restrictive or non-restrictive nature of a relative is dependent on the entities available in a discourse, as well as on the intention which may or may not be conveyed effectively by the speaker/writer.

For example, supposing we have one donkey, whose colour is unknown or whose colour is known to be black, sentence 125 is effectively non-restrictive – it either just adds new information or does nothing, respectively; if the speaker has indicated otherwise, then something is wrong. Alternatively, given more donkeys, some of which are known not to be black, the same sentence is *de facto* restrictive, whether or not the colour(s) of the non-black donkeys is(are) known, and again, if the speaker has indicated otherwise, something is wrong. Further, in this last case, if there are some donkeys which are not known not to be black, the relative is effectively both restrictive and non-restrictive, in that it rules out the non-black ones and imparts the information that (one of) the remainder is black.

"The donkey which is black eats the carrot" 125

The question, then, is of matching the effect actually achieved with pragmatic expectation. For example, we would not expect a reference which was intoned or punctuated as a restrictive not to reduce the candidate set. Equally, we would not expect a restrictive to apply to an already singleton set of candidates. In the event that such a mismatch occurs, we need to detect it, so we can take some warning action (or at least be prepared to do so in the event that the mismatch is not subsequently accounted for).

Therefore, it may be worthwhile to consider a view where the effect of the relative is derived from its pragmatic behaviour, and then to compare this with the effect inferred from intonation, puncuation or whatever. This accords with the assumption expressed in Chapter 1 that the speaker intends to make sense – so we simply add in the new information, enforce consistency, and then check that the speaker really did say what s/he apparently intended to.

4.2. George's Approach to Noun Phrase Post-Modification

George's approach to noun phrase post-modification is a new view which arises almost by necessity from the system's fundamental assumptions. As with all other kinds of reference, post-modified references are treated strictly incrementally. This forces us to take a rather degenerate, but nonetheless effective, approach to the dereferencing of such references. Note also that this line of reasoning, applied to the restrictive/non-restrictive distinction in adjectives, is implicit in the choice of open world non-contradiction (rather than closed world explicit consistency) for candidate selection.

The reasoning is as follows. Since we are working strictly incrementally, we must (by definition) always reach and analyse the definiteness and sort information associated directly with a reference by determiners, adjectives, and so on, contained in the main body of the noun phrase, before we reach that contained in post-modifiers. Suppose, then, that we follow the analysis approach already suggested as far as the end of the main body of the noun phrase – *ie* up to the start of the post-modifier. By this stage, we have produced a set (call it S_{before}) of candidate references for the main noun phrase embedded in the post-modified one. When we now proceed to analyse the post-modifier, using the same algorithm as before, we will refine the binding – and therefore eventually the entity token – in the same way as before with the information in the context extension (which may or may not be new). Call this new candidate set S_{after}.

If the set S_{before} is the same size as S_{after}, the post-modifier was not in effect restrictive, by definition – if it was intended to be so, it failed. Conversely, if S_{before} is strictly larger than S_{after}, the modifier was definitionally effectively restrictive, and, if it was intended to be non-restrictive, it failed. Either of these conclusions is reached in respect only of the referential properties of the phrase, and not through consideration of syntactic or pragmatic effects.

Why is this so? One convincing view of noun-phrase post-modification, arising from the definitions given above, is that a non-restrictive modifier adds information to the objects in the candidate set of the main body of the reference after that set has been found; and that the restrictive modifier contributes to the information in the noun phrase which, all together, subsequently selects the candidate set. This difference in timing seems to suggest that two different syntactic analyses are appropriate. The two conventional analyses for the head noun phrase of sentence 125 are shown in figure 8 – *a)* is the non-

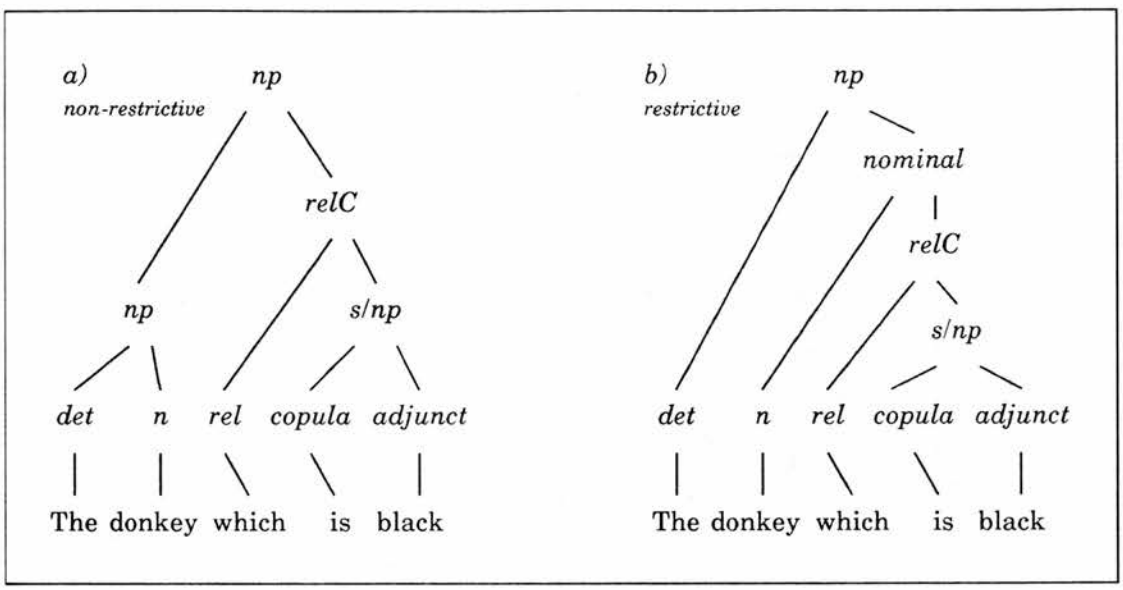


Figure 8: *Two possible syntax trees for "The donkey which is black"*

restrictive reading, *b)* is the restrictive.

Now, in a context of incremental reference evaluation, this distinction is immediately vacuous, because both analyses always reduce to the same thing: that the candidate set of a reference (or its intension if one so prefers) is initially (exhaustively) calculated and subsequently refined word by word as the parse proceeds through the post-modifier.

Therefore, I suggest that we do not need to work out from non-referential effects (*eg* intonation) which kind of post-modifier we dealing with. Following the reasoning process outlined above (of which an example is given in the next section) we will produce the same candidate set either way. In the George framework it is easy subsequently to compare the nature of the reference deduced from these non-referential sources with the *de facto* effect of the modifier by a simple textual comparison between the bindings and entity tokens defining S_{before} and S_{after}. We can then request clarification in the event that the intended and the actual effect of the modifier do not match. The point is, though, that we do not

need the syntactic/pragmatic information to work out the actual nature and effect of the modifier in the first instance.

(For completeness, we must also note that this view works, by default, when the noun phrase is introductory, be it so via an indefinite reference or (implicitly) via a definite reference whose candidate has been restricted to nothing. In this case the modifier can only be non-restrictive because there can only be one entity token in the candidate set.)

This argument leads me to conclude that, in an incremental context, we can view both restrictive and non-restrictive post-modifiers as functions of noun phrases, instead of viewing restrictives as functions of nouns, which is the view required by the more traditional analysis outlined above. I suggest that this is preferable because we do not need to decide in advance (or, strictly, at all) between non-restrictive and restrictive readings. Also (and this might be said to back up my suggestion), this new view provides a solution to the problem presented in the above traditional view by restrictives applied to single word noun phrases. An example of this is shown in sentence 126 which admits both restrictive and non-restrictive readings but cannot be subject to the syntactic analysis for restrictives given in Figure 8, above.

"Someone who is kind feeds the donkey." 126

4.3. An Example of Post-Modified Noun Phrase Processing

As an example of the treatment of restrictive relative clauses and of the issues involved in their processing (which are in fact the same as those of the relative in the first sentence of the running example, 106), let us now consider the effect of a third sentence – that shown below in 127 – on the George system. The example contains an uncontrovertibly restrictive relative, and for the first time shows Rule 6, stated formally here, working.

Let E be a candidate entity token for a reference, R, within a closed predicate containing other references R_i, not identical with R, then:

E is deleted from R's candidate set if:

1. **E is in a singleton binding with any of R_i; and**
2. **R's candidate set is not singleton.**

Rule 5: *Non-Co-Reference within Closed Predicates*

"He hates the person who beats him." 127

First, on input to the Parser of the word "He", the Parser passes the following set of references to the DeReferencer:

{ ref7!~male } 128

This will give rise to the (singleton) set of bindings shown in 129.

{ { { ref7!~male }, { e1, e2 }) } 129

Note here, again, that the set of entity tokens is very definitely a candidate set, for the following reason. Even though we already know that e1 specifies an entity which has the property "gender is male" (or something equivalent – the exact detail is not relevant here), e2 specifies an entity of sort "donkey", which has no implicit gender property. e2 may therefore at some later stage acquire a contradictory property (perhaps "gender is female" via a non-restrictive relative: "The donkey, who is female,...") and thus invalidate this binding. This, again, is the open world assumption.

Now, we might argue that, at this point, something like *focus* (as in [Grosz, 1977], [Sidner, 1979]) is required to decide between the two possibilities, and, indeed, this would be intuitively satisfactory. However, focus is outside our discussion here – though the George system has calls to focus functions included in part ③ of the top level algorithm, they currently do nothing.

The parse proceeds as before until we have a translation of the phrase "He hates the person who beats". This gives rise to the set of references shown in 130. Note again, here, that the "beat" information does not affect reference until the context extension of the reference is closed.

{ ref7!~male, [beats,var1,ref8~person!] ▸ ref8~person! } 130

As before, the non-closure of the subordinate clause causes George to ignore the relative information; and the DeReferencer calculates a set of bindings like this:

{ { { ref7!~male }, { e1, e2 }), ({ ref8~person! }, { e1, e3 }) } 131

Now, we are for the first time in this sentence in a position to try to apply Rule 5 (non-coreference in closed predicates). The rule relates to the set of bindings shown in 131, because e1 is held in common by both the bindings, and both the bindings appear in the

closed predicate representing the main clause of the sentence (shown in 132 – here, we see the DeReferencer consulting the Discourse Memory for the first time). However, at this stage, we have no means (other than focus, which we are deliberately not using) of deciding which (or, indeed, if either) of the two references should be bound to e1. The George DeReferencer, therefore, defers this problem, in the hope that the difficulty will be resolved before the end of the discourse.

[beats,var1,ref8~person!] ▸ [hates,ref8~person!,ref7~male!] 132

So now we go on to read the last word of the sentence, "him". In doing so, we close the context extension of the "person" reference, and so we are now able to attempt to use it to refine the reference of ref8. The new set of references output by the Parser is shown in 133.

{ref7!~male, [beats,ref10~male!,ref8~person!] ▸ ref8~person!, ref10~male!} 133

ref7 is dealt with as before. Next, the context extension of ref8 is considered, by matching the subordinate clause against previous closed predicates in the discourse. In effect, this candidate set refinement is only as incremental as [Mellish, 1985]'s approach, and less so than that of [Haddock, 1989], because of this closure constraint. However, if the final noun phrase of the context extension had contained more than one word, the effect would have been more incremental. Note also that the comparision involves an assumption of a world which is closed with respect to the discourse so far; only information already known can restrict the reference of a noun phrase. This is an approximation – in general, one would also wish to compare with *a priori* knowledge.

Matching, in this context, is a fairly loose term. At the simplest level it involves a simple first-order typed term unification between the context extension predicate "[beats,ref10...]" and *any other closed predicate appearing in the Discourse Memory*. Thus, the matching is made to cover information introduced in subordinate clauses as well as in main ones. The detail of the matching process is covered in Section 5.4.

The searching and testing outlined is encapsulated declaratively in Rule 6, which will also be expressed in algorithmic form in Section 5.4. It is necessary to consider both the reference information and the entity information mentioned in the rule because entity tokens often contain information from more than one reference, so they may be more restricted in sort than any given reference bound to them (so we need to look at the entity tokens, and not just the references). However, discovering the relevance of a preceding item of information depends on the reference of that information too, because of George's use of non-inconsistency instead of explicit consistency for candidate selection.

A post-modifier (or, in general, a clausal modifier) is considered to relate to information already presented in the discourse if and only if:

1a. There exists in the preceding discourse a sentence which expresses the information referred to by the context extension predicate – it has the same predicate and all corresponding references between the old and the new predicate are referentially consistent.

and 1b. The references in that preceding sentence are bound to entity tokens which are referentially consistent with the corresponding references in the context extension predicate.

or 2. Discourse world deduction allow us to generate an intermediate closed predicate from the context extension for which parts 1a and 1b above hold.

Rule 6: *Context Extension Matching*

Application of Rule 5 corresponds with that part of Haddock's mechanism relating to network *arc consistency*, and thence with the equivalent part of Mellish's system. As with node consistency, the reason it is different is George's more liberal, open world approach to the initial selection of candidates.

Returning to the current example, the effect of the context extension match is to rule out the woman as a candidate for the "person" reference, ref8: there is a relevant expression relating the man with beating the donkey in the Discourse Memory. This means that ref8 is now singleton bound to e1, the entity token specifying the man. Therefore, Rule 5 now excludes that entity token from the binding of ref7, the "He" reference. ref7, therefore, is now bound unambiguously, and correctly, to the donkey.

George's output after processing the third sentence is shown in Figure 9.

5. Some Loose Ends

5.1. Introduction

Three issues were raised in the preceding sections which have not yet been explained in full. To complete this discussion of simple reference we must deal, first, with George's approach to referential failure and its relationship to introductory reference, and, second,

```
>> He hates the person who beats him.
================================================
End of Sentence

   ** s:coref(ref2~unkind,ref1~man) ▸
            [beat(pres,pres,perf,act,indic),
               ref3~donkey~brown,
               ref1~man]
   ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]
   ** s:[beat(pres,pres,perf,act,indic),
               ref10~male!,
               ref8~person!] ▸
            [hate(pres,pres,perf,act,indic),
               ref8~person!,
               ref7~male!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey~male
ref7 is bound to e2~brown~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~brown~donkey~male

>>
```

Figure 9: *Continued George output for "He hates the person who beats him."*

with the necessity of maintaining consistency of reference throughout the entire discourse. Finally, we must cover the detail of the comparison process involved in dereferencing context extensions.

5.2. Introductory and Non-Introductory Reference

It has already been explained that, in George, simple indefinite references are always viewed as introductory – that is, they always give rise to new entity tokens. This is an approximation, to keep the theory manageable. A more general view is as follows. Referring expressions in English may be divided up along three partly independent dimensions. The first of these depends on the definite or indefinite nature of the expression (denoted in GRL by application of the definite operator, !). The second dimension is a less obvious distinction, which has in the past often been equated (for purposes of experiment) with definiteness: it depends on whether the information in the referring expression is "given" or "new" (in other words "already known" or "not already known"). The third dimension is buried deeper still in the referential behaviour of the expressions: it depends on whether the referring expression introduces a new entity into

the discourse or if instead it refers to an existing one. This third dimension might be further divided, depending on whether the entity introduced or referred to were generic or an individual; this aspect I will bypass for the purposes of simplicity here.

The point is that, as I said above, these three dimensions are actually partly independent. For example, while a definite singular noun phrase usually refers to an individual already existing in the discourse, it may introduce a new entity. Equally, an indefinite singular often introduces a new entity, but may well refer to one of an existing uniform set. The dimension of given/new-ness is related as well – first, if a referring expression is introductory, then the information in it must be new. If it is non-introductory then the information often will be already known, but in many cases it will not.

George uses a combination of these factors which aims to capture the reality of these aspects of human reference understanding more accurately than before. It is based on Crain and Steedman's Principle of Parsimony (see Chapter 2, Section 3), which states that analyses requiring less pre-supposition will be favoured over ones requiring more. First, then, the statement of the abstract mechanism. Note that this analysis is given in the abstract, ignoring issues of incremental parsing; these issues are incorporated in to the statement of the George DeReferencer top level algorithm in Section 2.1 of this chapter.

In GRL, all references are labelled definite or indefinite with the ! operator. In dereferencing we can always therefore case-split on the existence of that operator. In the event that a determiner or quantifier is ambiguous over definiteness, we can supply two different lexical entries for it – note that the coercion mechanism is inappropriate here, because this ambiguity is not (usually) resolved by subsequent words, but by referential consistency.

For the definite case, given a reference R, we proceed as follows.

If there is a set of entity tokens whose sorts are consistent with that of R and whose cardinalities are correct for reference by R, let this be the candidate set of R. If R is simple, the set is interpreted as a disjunction; if it is indexed, the set is interpreted as a conjunction (modulo focus, which I do not cover here). This choice requires no pre-supposition at all.

Otherwise, if there is a set of entity tokens whose sorts are consistent with that of R but whose cardinalities are not correct for reference by R, attempt (by the procedure described in Chapter 7) to find a (possibly singleton) subset of a set already existing in the discourse

which has the right cardinality for reference by R. This choice requires the weak pre-supposition that a set may be decomposed into subsets and/or individuals.

Otherwise, if there is no entity token whose sort is consistent with that of R, create one. This choice requires the much stronger pre-supposition that an entity exists which has not been explicitly introduced.

The indefinite case is simpler.

If there is an entity token specifying a set whose sort is consistent with that of R, attempt (by the procedure described in Chapter 7) to derive a (possibly singleton) subset of a set already existing in the discourse which has the right cardinality for reference by R. If there is such an entity token and the reference is felicitous this derivation will always be possible. Again, this choice requires the weak pre-supposition that a set may be decomposed into subsets and/or individuals.

If there is no suitable entity token, create one suitable for reference by R. Again, this choice requires the strong pre-supposition that an entity exists which has not been explicitly introduced.

As I said before, the indefinite case has been simplified in George, to the assumption that indefinites are always introductory. This will cause a (not unsoluble) problem when we come to deal with dependent set references, where indefinites are in the (conventional) scope of quantifiers, in Chapter 7. Similar questions arise when indefinite reference appears in negated expressions, and in commands and questions. In Chapter 9, I will suggest possible extensions to GRL to account for these.

Finally, the sequence of tests and actions specified above begs one more question about George. In Chapter 3, I stated that there is a "suicide pact" between the George Parser and the DeReferencer, such that if one failed, so did the other. Where then, has this pact gone? It would seem from the desciption above that it is impossible for the DeReferencer to fail, in particular in the case of a definite reference whose candidate set has become empty.

The answer to this question is in two parts. First, recall from the top level algorithm that there is no option for replacement of an entity which has somehow been ruled out for binding to the indefinite reference which introduced it. In this case, the system, correctly, would fail.

The more interesting part of the question depends on the choice of action when the candidate set of a definite reference become empty. Here, I have stated that a such a reference should be viewed as introductory, which is a view at odds with other incremental approaches where failure of reference is sometimes used to rule out syntactic parses. Why, then, do I not take that approach in George? Because I make here an assumption that the speaker/writer intends to be felicitous (see Chapter 1) and that the purpose of a system like George is not to make correctness judgments on his/her use of reference, but to attempt to "understand" it. Since definite references can be introductory, it is incorrect to rule out that possibility. (Having said this, one might certainly wish to build a theory which was rather more selective than George about when it allowed such introduction – as such, George is an approximation.) Also, other factors (*eg* discourse world inference) might cause failure of the DeReferencer in a more complete implementation.

Another use of reference is the distinction between restrictive and non-restrictive relatives. [Haddock, 1989] gives a treatment of relatives which relies on the emptiness or over-fullness of a candidate set, and thence failure of the translation which gave rise to it, to disambiguate. Essentially, he calls upon something like the Principle of Referential Failure (see Chapter 2, Section 3) to do the disambiguation. I will argue in Chapter 8 that Haddock's method is incorrect. In George, the Principle of Parsimony (which subsumes that of Referential Failure) is built in to the order of the tests in the dereferencing algorithm. Further, there is no syntactic difference between restrictives and non-restrictives. Thus, disambiguation in the sense of failing possible translations is simply not appropriate.

The point is that GRL is adaptable. Different possible syntactic parses will often be written conflated in one ambiguous expression, and not enumerated separately. This just does not fit in with the idea of referential failure causing (*eg*) backtracking or failure of a possible translation in a breadth-first system. Rejection, for example, of the restrictive/non-restrictive distinction in any sense other than as a retrospective check means that the actual referential behaviour of a given noun phrase will never affect the syntactic form of the sentence in which it is embedded.

I suggest, then, that this view of interaction between parsing and reference analysis (subject to the comments regarding approximation, above) is a realistic one, and that it fits in both with my own (reasonable) assumption that a speaker/writer intends to make sense and with the Principle of Parsimony of Crain and Steedman, which is justified in the original sources (*eg* [Crain & Steedman, 1985]).

5.3. Consistency Maintenance in George

Because George is a strictly incremental system, it is necessary on some occasions to ensure that old material is updated to make it consistent with new. Consider, for example, the discourse in 134.

"Jim and Fred own a donkey. The thin man beats the donkey. Jim is fat." 134

After the second sentence, the candidate set for "The thin man" contains tokens specifying Fred and Jim (recall that non-inconsistency is required, not explicit consistency, for candidate set membership). Only when the information that Jim is fat is added to Jim's entity token, in the third sentence, can we rule it out as a candidate for reference by "The thin one". When we have done so, we have a singleton singular binding, so we must allow the sort information in the reference to pass to the entity token, as usual. This constitutes an inference from the discourse that Fred is thin. This process involves manipulation of references older than those in the current utterance, which is not the case in the rest of the top-level dereferencing algorithm, except for context extensions, more of which below.

In general, this process must happen continuously (or at least continually) as the references and entity tokens in the George system become more refined – this is the part of George which aims to resolve referential ambiguity in context larger than the sentence in which it arises. It can be implemented as a further parallel process within each basic George system, or (as in the current implementation of George) as a procedure which is called each time the Entity Tokens and Bindings databases are updated.

One rule in particular which is applied in this way is Rule 6, which precludes non-reflexive referring expressions within a clause from co-referring. Note that the rule only takes effect to rule out a candidate from one reference when there is no alternative to that candidate being bound to a different reference – in weaker cases (*eg* when it is not known which of a pair of references really is bound to a candidate) the rule has no effect.

5.4. Context Extension Comparison and Inference

5.4.1. Using Information in Context Extensions

As I mentioned before, in attempting to calculate the reference of context extended references, George performs simple inference from the translation of the preceding discourse. The pattern matching algorithm it uses to find suitable information from which to do so is an area requiring further work, but, as in other such cases, enough has been

done to show that the idea is sound. In this section, I will explain the algorithm, give a simple example of its use, and show how it can resolve attachment ambiguity, in a way comparable with that of [Winograd, 1972]'s system, SHRDLU.

It is important to understand that, while making an open world assumption with respect to entities in the discourse world, George assumes a closed world with respect to situation information in the discourse. Thus, only the translations of utterances from the discourse contribute to the understanding of context extensions, and abduction is not allowed. Such a rule is clearly not general, because it precludes *a priori* knowledge about situation information common to both speaker and hearer from affecting reference analysis. Nevertheless, it is a useful and safe simplifying approximation – in any case, *a priori* shared knowledge can always be simulated by initial discourse. Note also that the closed world with respect to which the context extensions are analysed is continually increasing in size as the discourse proceeds, and, in particular, that context extensions, once dereferenced, become part of that world.

5.4.2. Comparing Context Extensions with Discourse

In order to ascertain which (if any) of earlier utterances may be used as a basis for inference about the context extended reference (which I shall call being *relevant*), George follows the following procedure, first using the context extension itself, and then again using other closed predicates deducible from it via backward-chained discourse world inference (backward-chaining is used because it leads to a smaller search space than forward chaining from each sentence in the discourse, as in [Mellish, 1981]). This algorithm is an alternative statement of Rule 5.

Some more terminology: One of the references in a context extension always appears on both sides of the ▸ operator (because of the way context extensions are constructed). This reference is the one which is being refined by the information in the context extension; I shall call it the *topic* of the extension. Other references are then *non-topical*. An earlier expression being compared is a *precursor*. Note that precursors are considered in the reverse order of that in which they were introduced into the system. This expresses the assumption that new information is likely to be more relevant than old.

1. Is the predicate name in the context extension the same as in the precursor? If not, the precursor is not relevant. (Note that this is a very coarse approximation – at least temporal reasoning, and ultimately reasoning about such issues as the habitual reading of continuous tenses must be included here for a complete coverage.)

2. Is the predicate arity the same? If not, the precursor is not relevant. (This is again an approximation. For example, we might wish to include an optional instrument as a third argument in the translation of some verbs, so a precursor whose predicate's arity was different from that of a context extension omitting that instrument might still be relevant.)

3. Are the references in the precursor consistent in sort and number with those in the context extension? If so, the precursor is relevant. If not, it is irrelevant. This is a check to determine whether the precursor *could* refer to the same discourse entities as the context extension.

Once the precursor is shown to be relevant to the context extension, George computes the intersection between the candidate set of each reference in the context extension and the set of entity tokens to which the corresponding reference in the precursor is bound. In each case, this intersection is the new candidate set for the appropriate reference in the context extension. If any candidate set is empty, the topic of the context extension cannot be (one of) the candidate(s) of the corresponding reference in the precursor. George therefore removes it from the candidate set of the topical reference, and proceeds in the attempt to find a precursor which is relevant. In the event that the candidate set of the topical reference is empty, a new entity token must be synthesised, as usual in George. Note that the analysis presented here lacks in that it does not allow for the case where more than one precursor is relevant.

Note also that this is one point at which the division in George between references and discourse entities becomes crucial. Without it (for example, if entities were substituted for references in the discourse as soon as bindings were established) the test procedure above could not work. This is because both the notion of relevance and of candidate set intersection determine whether a context extension is referentially consistent with a precursor. For example, if a context extension is relevant to a precursor, but only one (of the two, say) references in it refers appropriately then the topic context extension is wrongly bound. This will become clear in the example, below. The method is more complicated than the uniform constraint satisfaction used by Mellish and Haddock because George picks candidates by non-contradiction and not explicit consistency.

5.4.3. An Example of the Use of Context Extension Information

Let us consider an example of such a search. Suppose we have the discourse, translation, and bindings shown in 135a, b and c – the binding for ref8 having been refined only by sort

and not by context extension at this stage. We must now analyse the effect of the context extended reference in the final sentence; to do so, we perform the matching procedure outlined below. If the post-modifier giving rise to the context extension is non-restrictive, we will simply fail to match with a previous utterance; if we wish, we can check this against any pragmatic information regarding restrictiveness in the modifier.

"Jim owns a donkey.	(i)	
Jim owns a house.	(ii)	
Liz owns a dog.	(iii)	
Fred feeds the dog.	(iv)	
Liz hates the donkey.	(v)	
Liz beats the animal which the man owns."	(vi)	135a

[owns,ref2~donkey,ref1~Jim!]
[owns,ref4~house,ref3~Jim!]
[owns,ref6~dog,ref5~Liz!]
[feeds,ref8~dog!,ref7~Fred!]
[owns,ref10~animal!,ref11~man!] ▸ [beats,ref10~animal!,ref9~Liz!] 135b

⟨{ref1~Jim!},{e1~Jim}⟩, ⟨{ref2~donkey},{e2~donkey}⟩,
⟨{ref3~Jim!},{e1~Jim}⟩, ⟨{ref4~house},{e3~house}⟩,
⟨{ref5~Liz!},{e4~Liz}⟩, ⟨{ref6~dog},{e5~dog}⟩,
⟨{ref7~Fred!},{e6~Fred}⟩, ⟨{ref8~dog!},{e5~dog}⟩,
⟨{ref9~Liz!},{e4~Liz}⟩,
⟨{ref10~animal!},{e2~donkey, e5~dog}⟩, †
⟨{ref11~man!},{e1~Jim, e6~Fred}⟩⟩ † 135c

All of the reference evaluation in this example is trivial (modulo some assumptions about proper names which would only cloud the issue here), except for that of ref10 and ref11 (marked † in the bindings in 135). ref10 is the topic of the context extension in 135b.

Assume we have parsed this rather boring discourse up to and including the word "man" in the last sentence. When the word "owns" appears, the context extension becomes closed, and can therefore be used for reference evaluation. The extension is compared with the five precursors (the foregoing sentences) in the following way. First, sentences (iv) and (v) (about dog-feeding and hatred, respectively, and not about ownership) are ruled out by test 1 in the procedure above. They are irrelevant at the coarsest possible level. Sentences (i), (ii), and (iii) all pass both of the first two relevance tests.

Now we apply the third test to the three remaining candidate precursors. They are all consistent in number with our context extension, because there are only only singular

references in this example. However, sentence (ii) is ruled out, because "house" is inconsistent with "animal". Sentence (iii), similarly, is ruled out because "Liz" is inconsistent with "man". Sentence (i), though, passes test 3, so we are left with a single relevant precursor. Now, replacing each of the candidate sets in the context extension with the intersection between itself and those of the matching reference in the precursor, we are left with the bindings for the whole discourse shown in 136, which are correct.

$$\{\langle\{\text{ref1}^{\frown}\text{Jim!}\},\{\text{e1}^{\frown}\text{Jim}\}\rangle,\langle\{\text{ref2}^{\frown}\text{donkey}\},\{\text{e2}^{\frown}\text{donkey}\}\rangle,$$
$$\langle\{\text{ref3}^{\frown}\text{Jim!}\},\{\text{e1}^{\frown}\text{Jim}\}\rangle,\langle\{\text{ref4}^{\frown}\text{house}\},\{\text{e3}^{\frown}\text{house}\}\rangle,$$
$$\langle\{\text{ref5}^{\frown}\text{Liz!}\},\{\text{e4}^{\frown}\text{Liz}\}\rangle,\langle\{\text{ref6}^{\frown}\text{dog}\},\{\text{e5}^{\frown}\text{dog}\}\rangle,$$
$$\langle\{\text{ref7}^{\frown}\text{Fred!}\},\{\text{e6}^{\frown}\text{Fred}\}\rangle,\langle\{\text{ref8}^{\frown}\text{dog!}\},\{\text{e5}^{\frown}\text{dog}\}\rangle,$$
$$\langle\{\text{ref9}^{\frown}\text{Liz!}\},\{\text{e4}^{\frown}\text{Liz}\}\rangle,$$
$$\langle\{[\text{owns,ref10}^{\frown}\text{animal!,ref11}^{\frown}\text{man!}] \blacktriangleright \text{ref10}^{\frown}\text{animal!}\},\{\text{e2}^{\frown}\text{donkey}\}\rangle, \quad \dagger$$
$$\langle\{\text{ref11}^{\frown}\text{man!}\},\{\text{e1}^{\frown}\text{Jim}\}\rangle\} \quad \dagger \qquad 136$$

5.4.4. Resolving Attachment Ambiguity

This method also allows us to deal with the problems addressed by [Winograd, 1972] and [Haddock, 1989] where dependent references, ambiguous in isolation, have unambiuous meanings when taken together. Winograd's well-known example was set in the blocks world. Given some boxes, some blocks, and a table, the problem is to work out the correct reading of "...put the block in the box on the table.". George covers this class of examples, as follows. (It will only be necessary here to show how one of the two readings is detected – the other is simply the converse.) The possible translations of the problem sentence are shown in 137. Note that the (unsatisfactory) representation of the position argument of "put" and "keep" below was not formally defined as part of GRL in Chapter 4. This notation is included here as a stop gap, since I have not attempted to find a "correct" way to represent this kind of verb argument. One possible way might well be a situation semantics-like approach of introducing a new reference to a position and a context extension associating it with the reference in the introductory prepositional phrase:

"Jim puts the block in the box on the table" 137a

$[\text{in,ref3}^{\frown}\text{box!,ref2}^{\frown}\text{block!}] \blacktriangleright [\text{puts,[on,ref4}^{\frown}\text{table!}],\text{ref2}^{\frown}\text{block!,ref1}^{\frown}\text{Jim!}]$ 137b

$[\text{on,ref4}^{\frown}\text{table!,ref3}^{\frown}\text{box!}] \blacktriangleright [\text{puts,[in,ref3}^{\frown}\text{box!}],\text{ref2}^{\frown}\text{block!,ref1}^{\frown}\text{Jim!}]$ 137c

Suppose that we have a context arising from a preceding discourse like 138 (a is the English, b the GRL and c the bindings). Suppose also that we can infer that if something is kept in something it is in that something; this kind of inference is not really a feature of the George system, although it is possible, and works in the implementation:

"Jim owns a block. He keeps it in a box. He owns a table." 138a

[owns,ref6~block,ref5~Jim!]

[keeps,[in,ref9~box],ref8~it!,ref7~male!]

[owns,ref11~table,ref10~Jim!] 138b

{⟨{ref6~block},{e2~block}⟩,⟨{ref5~Jim!},{e1~Jim}⟩,
⟨{ref9~box},{e3~box}⟩,⟨{ref8~it!},{e2~block}⟩,⟨{ref7~male!},{e1~Jim}⟩,
⟨{ref11~table},{e4~table}⟩,⟨{ref10~Jim!},{e1~Jim}⟩} 138c

What happens, then, as we analyse the input in 137? The first two words, "Jim puts", are parsed and dereferenced as explained before – nothing particularly interesting happens. When the third word, "the", is read, the translation bifurcates, uninterestingly, into two possible readings for singular and plural (which arguably should be represented once, adaptably, anyway). This very local ambiguity is resolved at the next word, when the plural reading is rejected. At this stage, we have the following adaptable partial translation. The adaptability is in the representation of a post-modified noun phrase, which will be coerced to a plain noun phrase in one possible reading, later.

λvar2.λvar1.[var2,ref2~block!] ▸ [puts,var1,ref2~block!,ref1~male!]
{⟨{ref1~male!},{e1~Jim}⟩,⟨{ref2~block!},{e2~block}⟩} 139

When we read the next word, "in", something more interesting happens. The new word can combine with either the translation above as written, with the object noun phrase post-modified, or after it is coerced to an unmodified phrase. This is because the prepositional phrase can be interpreted either as a noun phrase modifier or as an argument filler for "puts". We now have the following translations (remembering that George does not use information in unclosed context extensions for dereferencing:

λvar3.λvar1.[in,var3,ref2~block!] ▸ [puts,var1,ref2~block!,ref1~male!]
{⟨{ref1~male!},{e1~Jim}⟩,⟨{ref2~block!},{e2~block}⟩} 140a

λvar4.[puts,[in,var4],ref2~block!,ref1~male!]
{⟨{ref1~male!},{e1~Jim}⟩,⟨{ref2~block!},{e2~block}⟩} 140b

Again, the next word, "the", introduces uninteresting local ambiguity. It is resolved by reading the associated noun, "box". This, and the associated dereferencing, leaves us with two new possible adaptable readings in 141. In particular, note that the inner context extension in reading a is closed; context extension matching and inference can therefore proceed on it. When it is compared with the existing discourse, the only relevant expression is that inferred from the fact that Jim keeps his block in his box: that the block is in the box. Now, the candidates of the corresponding references in these two statements

are identical and singleton. Thus, the matching process causes no change – as we would wish, because the bindings were correct in the first place.

λvar5.λvar1.[var5,ref3~box!] ▸
[in,ref3~box!,ref2~block!] ▸ [puts,var1,ref2~block!,ref1~male!]
{⟨{ref1~male!},{e1~Jim}⟩,⟨{[in,ref3~box!,ref2~block!] ▸ ref2~block!},{e2~block}⟩,
⟨{ref3~box!},{e3~box}⟩} 141a

λvar5.[var5,ref3~box!] ▸ [puts,[in,ref3~box!],ref2~block!,ref1~male!]
{⟨{ref1~male!},{e1~Jim}⟩,⟨{ref2~block!},{e2~block}⟩,⟨{ref3~box!},{e3~box}⟩}141b

Next, we read the word "on". The result is now three ways ambiguous, because, aside from the two possible readings available from the whole sentence, there is the possibility that the "on" phrase modifies "the box", and, with reading 141a, it is not possible to say at this stage that this is not the correct interpretation. For the purposes of example here, I will omit this spurious reading (which is ruled out by George at the end of the sentence). Next, then, we consider these two partial readings, with no change in reference.

λvar6.[in,ref3~box!,ref2~block!] ▸ [puts,[on,var6],ref2~block!,ref1~male!]
{⟨{ref1~male!},{e1~Jim}⟩,⟨{[in,ref3~box!,ref2~block!] ▸ ref2~block!},{e2~block}⟩,
⟨{ref3~box!},{e3~box}⟩} 142a

λvar6.[on,var6,ref3~box!] ▸ [puts,[in,ref3~box!],ref2~block!,ref1~male!]
{⟨{ref1~male!},{e1~Jim}⟩,⟨{ref2~block!},{e2~block}⟩,⟨{ref3~box!},{e3~box}⟩}142b

Finally, we read the last noun phrase, "the table". This results in the following translations and bindings before context extension matching and inference has been applied.

[in,ref3~box!,ref2~block!] ▸ [puts,[on,ref4~table!],ref2~block!,ref1~male!]
{⟨{ref1~male!},{e1~Jim}⟩,⟨{[in,ref3~box!,ref2~block!] ▸ ref2~block!},{e2~block}⟩,
⟨{ref3~box!},{e3~box}⟩,⟨{ref4~table!},{e4~table}⟩} 143a

[on,ref4~table!,ref3~box!] ▸ [puts,[in,ref3~box!],ref2~block!,ref1~male!]
{⟨{ref1~male!},{e1~Jim}⟩,⟨{ref2~block!},{e2~block}⟩,
⟨{ref3~box!},{e3~box}⟩,⟨{ref4~table!},{e4~table}⟩} 143b

For reading a, context extension matching has already proceeded to completion; it causes no further change in the bindings. For reading b, the story is more complicated. This is the reading where the block is placed into some box already on the table, and is not (necessarily) already inside one. Winograd's (closed world) view is that this reading is infelicitous here, because there is no box on the table. In an open world, however, this is not a correct solution – it is so only if we know there is no other box. Restricted use of a closed world assumption would allow George to emulate SHRDLU here.

In George, context extension matching and inference proceeds as follows. The discourse is searched for a relevant precursor, and none is found: in the first instance, this implies that the context extension is new information that the box was on the table all along. However, there is another element to consider, which has not been built into George as presented here. There is a simple inference to be made based on the semantic requirement of "put" that the block cannot be put into a box which already contains it. On this basis, one or both of the candidate sets in the context extension must be wrong. In either case, if we remove the offending entity token, we are left with one or two empty binding(s); George's standard response to this is to create appropriate new entity tokens, on the supposition that the references in the empty bindings are therefore introductory. We can apply preferences to these possibilities, as I suggested before. In particular, due to the nature of relative clauses, it is more likely that the topic of a context extension will be introductory than that non-topical references therein will be so. Equally, readings involving presupposition of one new entity will be preferred (by Parsimony) over readings involving two or more. Taking this course, we have one most likely possibility for this second attachment structure: there must be a different box, resting on the same table. This constitutes a correct alternative reading of the sentence, which was ruled out of Winograd's analysis by his closed world assumption. The analysis, again, accords with the Principle of Parsimony; minimal presupposition is being imposed by the removal of just the topical reference's entity token.

We now, finally, have these two readings. The new box is specified by e5 in reading b.

[in,ref3~box!,ref2~block!] ▸ [puts,[on,ref4~table!],ref2~block!,ref1~male!]
{⟨{ref1~male!},{e1~Jim}⟩,⟨{[in,ref3~box!,ref2~block!] ▸ ref2~block!},{e2~block}⟩,
⟨{ref3~box!},{e3~box}⟩,⟨{ref4~table!},{e4~table}⟩} 143a

[on,ref4~table!,ref3~box!] ▸ [puts,[in,ref3~box!],ref2~block!,ref1~male!]
{⟨{ref1~male!},{e1~Jim}⟩,⟨{ref2~block!},{e2~block}⟩,
⟨{ref3~box!},{e5~box}⟩,⟨{ref4~table!},{e4~table}⟩} 143b

What is more, viewing both translations, now, as they proceed in parallel, the first requires (in a naïve sense) less presupposition than the second (because it introduces no new entities). As such, it can be favoured by yet another application of the Principle of Parsimony, if a mechanism to allow this were built on top of George. Alternatively, use of the closed world assumption is equivalent to enforced maximal parsimony.

This example has shown, then, that the view presented here of noun phrase post-modification need not detract from the ability of a computational system to solve problems of attachment ambiguity raised by Winograd and Haddock.

6. Indexed Reference

6.1. Introduction

The idea of *indexed reference* gives GRL the ability to represent plural utterances which refer to sets. An indexed reference takes the same form as a simple (*ie* non-indexed) reference, with the addition (as specified formally in Chapter 4) of a quantified variable, or *index*, ranging over $\mathbb{N}$, whose association with a particular reference symbol is indicated by the *strong* and *weak index application* operators, $\otimes$ and $\times$. The distinction between the two index operators will be explained in the next chapter; for the moment the two can be viewed as the same. The operator symbols in a sense reflect the semantics of the operation, which, in the simplest cases, is not unlike the elaboration of cross-products of sets. The value of an index may be unbounded above or it may have an upper bound. This upper bound may be explicit (*eg* introduced by a number quantifier) or implicit (*eg* deduced from the entity tokens available for binding to a definite reference). The GRL expressions in 144 are examples of GRL representations of English plural and quantified noun phrases and sentences. Precedence is such that, in 144h, the quantifier dominates the context extension operator.

"The men"	∀ind1.ref1ˇman!×ind1	144a
"Some men"	∀ind1.ref1ˇman×ind1	144b
"Two donkeys"	∀ind1<2.ref1ˇdonkey×ind1	144c
"The two donkeys"	∀ind1<2.ref1ˇdonkey!×ind1	144d
"A man owns some donkeys"	∀ind1.[owns,ref2ˇdonkey×ind1,ref1ˇman]	144e
"The men own the donkeys"	∀ind1.∀ind2.[owns,ref2ˇdonkey!×ind2, ref1ˇman!×ind1]	144f
"Every man owns a donkey"	∀ind1.[owns,ref2ˇdonkey,ref1ˇman!⊗ind1]	144g
"Every man who owns a donkey beats it"	∀ind1.[owns,ref2ˇdonkey,ref1ˇman!⊗ind1] ▸ [beats,ref3ˇit!,ref1ˇman!⊗ind1]	144h

In Chapter 4, I explained that it is the ranging of the index across (a contiguous subset including 0 of) $\mathbb{N}$ which carries the force of the quantifier to the quantified reference. The reason for choosing what might at first seem to be an obstruse notation is that that notation, as it were, divides the quantification into two parts – there is the ranging over $\mathbb{N}$, and then there is the effect of this on the references. In itself, this is not very interesting, but, in order to reason in full generality about set reference, we will want to consider

interaction between references. This will not, in general, affect the behaviour of the quantification, but it will affect the way the quantification affects the references. GRL allows us to manipulate these aspects independently. An important example of the kind of interaction I mean here is that between quantifiers conventionally expressed by manipulating quantifier scope. I will explain the details of all this in Chapter 7.

For the moment, the use of indices ranging across $\mathbb{N}$ gives an indication of the intuition involved here. We can represent a finite or countably infinite homogenous collection of entities by a single entity token, thus making reference to the totality of a set (represented by a single entity token) procedurally equivalent in the George system to reference to an individual (or singleton set, also represented by a single entity token). This analysis rather bypasses the question of whether a predication of a set is distributive or not. I suggest, however, that the distinction is often irrelevant in discourse processing, and this ambiguity is often left unresolved. In particular, distributive readings can be derived from collective ones by spreading a relation across the members of a set, though this is not always true the other way round. Therefore, a representation may be viewed as being ambiguously distributive or collective, so long as there is a distinguishable distributive representation to which it may be coerced. I will propose such a system in Chapter 7.

Let us now proceed with our running example, to demonstrate the behaviour outlined above. At this rather simple stage, it will be seen that the behaviour of the DeReferencer, beyond the inclusion of the index notation, is exactly the same for indexed references as for simple ones, which is as we would like, given our step-by-step approach to the design of the George DeReferencing system. The discourse so far is shown in 145.

"A man who is unkind beats a donkey.
A woman feeds the donkey.
He hates the person who beats him." 145

6.2. Introductory Indexed Reference

Suppose that we now input sentence 146 to the Parser.

"The woman gives the donkey some carrots." 146

The reference to "The woman" and "the donkey " here are dealt with in exactly the same way as "the donkey" in the second sentence of 145, with corresponding results: we arrive at two simple references bound to the existing entity tokens specifying the woman and the donkey respectively. When we reach the final noun phrase, though, we have a different

situation. The output from the Parser to the DeReferencer after reading "some" is as shown in 147. Note that the alternative non-introductory reading of "some" (as in "some of the...") has been omitted here, as before, because it will lead to a failed translation and therefore serve no purpose in the example.

$$\{ \text{ref11}\tilde{}\text{woman!}, \text{ref13}\tilde{}\text{donkey!}, \forall \text{ind5}.\text{ref15} \times \text{ind5} \} \qquad 147$$

Following the top level algorithm (which does not refer to indexing at all), this will cause the creation of a new binding, and a new (as yet unsorted) entity token to be bound to ref15. It would be labouring a point to show again in detail how the sort information from "some carrots" arrives at the binding and then propagates to the entity token because of the singleton nature of introductory bindings. The full detail can be seen in Appendix B; and the final analysis is shown in figure 10.

```
>> The woman gives the donkey some carrots.
================================================
End of Sentence

 ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),
           ref3~donkey~brown,
           ref1~man]
 ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]
 ** s:[beat(pres,pres,perf,act,indic),ref10~male!,ref8~person!] ▸
        [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]
 ** s:∀ind5.[give(pres,pres,perf,act,indic),
              ref13~donkey!,
              ref15~carrot×ind5,
              ref11~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey~male
ref7 is bound to e2~brown~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~brown~donkey~male
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot

>>
```

Figure 10: *Continued George output for "The woman gives the donkey some carrots."*

This example shows, then, how plural introductory references are made. The procedure, formally stated in Rule 7, is exactly the same as for singular introductory references, the

A new indefinite indexed reference is always singleton bound to a new entity token created for that purpose.

Rule 7: *Indexed Indefinite Reference*

difference being that the binding contains a quantifier and index in its references. Now, with Rule 4 (number consistency), we have here all the mechanism we need to talk about sets of the pragmatic kind covered by George. The next sentence in our example shows how non-introductory indexed references fit equally uniformly into our existing mechanism.

6.3. Non-Introductory Indexed Reference

Let us now examine the behaviour of the George system when we input sentence 148 in context of our running example (Figure 10).

"The donkey eats the carrots." 148

George's behaviour for the first three words is exactly as was demonstrated before. When we input the second "the", the Parser passes the following set of references to the DeReferencer (again, ignoring the unsuccessful translation arising from the singular reading of "the").

{ ref16~donkey!, ∀ind7.[var62,ref19!×ind7] ▸ ref19!×ind7 } 149

(The context extension here arises from the ambiguous representation of determiners and quantifiers introduced in Chapter 5. At the end of this sentence, we will be left with an item of syntactic category s/nmod. This will be coerced to s when the end of the sentence has been flagged by the full stop, because of the "NP-Post-Modifier" coercion. The corresponding semantic change is the removal of the context extension.)

Note that the quantifier is very much part of the reference, and not a separate part of the expression containing it. It expresses the set nature of the reference. However, the quantifier makes no difference to the treatment of the simple definite reference to "the donkey" and its binding, which proceeds as described before for singular non-introductory

references. When we analyse the determiner of the second noun phrase, we arrive at a set of bindings (omitting those from previous sentences for clarity) like this. This was an application of Rule 8.

The initial candidate set of a new indexed definite reference always initially contains all the entity tokens in the system which are both sort consistent in GRL and number consistent in George (Rule 4) with the reference.

Rule 8: *Indexed Definite Reference*

$$\{\langle\{\text{ref16}^{\smile}\text{donkey!}\},\{\text{e2}\}\rangle,$$
$$\langle\{\forall\text{ind7}.[\text{var62},\text{ref19!}\times\text{ind7}] \blacktriangleright \text{ref19!}\times\text{ind7}\},\{\text{e4},\text{e3},\text{e1}\}\rangle\} \qquad 150$$

Finally, we add the sort information "carrot" to ref19, which becomes inconsistent with e1 and e3 as a result. The binding is now therefore a singleton binding (containing e4), and so the sort information in the reference passes along the binding to the entity token. The context extension is now coerced away because of the sentence ending, and we are left with the bindings for this sentence shown in 151.

$$\{\langle\{\text{ref16}^{\smile}\text{donkey!}\},\{\text{e2}\}\rangle,\langle\{\forall\text{ind7}.\text{ref19}^{\smile}\text{carrot!}\times\text{ind7}\},\{\text{e4}\}\rangle\} \qquad 151$$

This example shows that, by viewing the reference as binding to some abstracted representation (*viz* the entity token) of the things to which it refers in the discourse world, rather than to those things themselves, we can defer the issue of which of them are actually included in the referent set, on a demand-driven basis. When (or if) we attempt to find which discourse entities are specified by this abstract representation we have information regarding number readily available, in the binding, which will help us in doing so.

This idea of binding to abstractions rather than to individuals captures exactly the intuition that when we refer to sets, even in complicated discourse involving subsets and unions, we do not in general need to know number information (though we may of course use it to restrict our search space, if it is supplied). George's process of finding candidate sets for bindings does not refer to number information except in the sense of consistency checking defined in Rule 4. This consistency check can only exclude incorrect bindings; it does not contribute to finding them, and thus prunes the search tree, as we would wish.

6.4. Indexed Reference to Unions of Sets

The very simple view of set reference presented above works well enough for discourses which involve sets unless those sets are subsets or unions of sets. Since real discourse is likely to include such more complex references frequently, we must now consider how George deals with them.

Let us first consider references to unions of existing sets. This occurs, for example, when we introduce two sets of people, maybe males and females, and then refer to the union with, for example, "They". As such, unions of sets can only arise in George through non-introductory reference, since introductory references are always bound to exactly one entity token. As an illustration, I will return to our running example, reproduced in 152, and consider the input utterance in 153.

"A man who is unkind beats a donkey.
A woman feeds the donkey.
He hates the person who beats him.
The woman gives the donkey some carrots.
The donkey eats the carrots." 152

"The people ride the donkey." 153

Taking, as before, only the successful (*viz* plural) reading of the definite article, on reading the first word the Parser outputs a set of references like this:

$$\{\forall ind8.ref21! \times ind8\} \qquad 154$$

which, according to the original top-level algorithm we followed before, gives rise to the binding shown in 155.

$$\{\langle\{\forall ind8.ref21! \times ind8\}, \{e1, e2, e3, e4\}\rangle\} \qquad 155$$

Here, then, as before, we have a candidate set which is a superset of the set we eventually wish to achieve. Note that there is no difference between this and the singular definite case where we require that the candidate set be singleton in the final analysis, because, even in that singular case, George allows non-singleton candidate sets on the basis that they represent ambiguous references which may be subsequently resolved. (Note again the difference in interpretation of the candidate sets of singular and plural references –

the former as a disjunctive set of candidates, the latter as a conjunctive set of correct answers.)

On addition of the head noun to the input, the set is restricted to the two people by application of the "person" sort, which is inconsistent with carrots and donkeys, leaving us with the binding shown in 156.

$$\{\langle\{\forall ind8.ref21\ \text{-}person! \times ind8\}, \{e1, e2\}\rangle\} \qquad 156$$

We now have the correct candidate set for this example, the full output for which is shown in Appendix B. A summary, including output for the previous example, sentence 148, is given in Figure 11.

Applying this approach to the cases where the candidate set is fully specified by the reference yields a correct set of entity tokens for the binding. In cases where the candidate set is not fully specified (*ie* the reference is ambiguous), the result achieved, as elsewhere in George, has the same status as candidate set results in [Webber, 1979] – the candidate set represents what is *available for reference*. We will therefore subsequently need to apply checks to see which of its elements are really bound to the reference. (Note, though, that we apply the same process to all references. Ambiguity or otherwise is emergent from the result.) For example, suppose the subject noun phrase in 153 had been replaced with a less specific reference, as in 157.

"They ride the donkey." 157

In this case, we start off with the same binding as shown in 155, but, because there is no sort information in the reference, the binding is not refined until the end of the second noun phrase. At this point, the appearance of a singleton simple binding involving the donkey allows its entity token to be ruled out of the indexed reference, by Rule 6 (non-coreference in closed predicate). However, there still remains one clearly incorrect candidate, the carrots. I suggest that the only real answer to this problem is discourse world inference (*eg* $\forall x.carrot(x) \Rightarrow \neg \exists y.rides(y,x)$); as such, the candidate set is serving as a starting point for such inference, and thus restricting the potential explosion of inferences necessary. This inference might be more neatly expressed by the specification of selectional restrictions on predicates, as explained in Chapter 9, in which case there is no question of search for a relevant inference rule. Such a means of expression fits neatly around the existing George approach to composition of noun phrase semantics, which was explained in Chapter 5. This kind of inference was possible in [Mellish, 1981]'s program.

```
>> The people ride the donkey.
================================================
End of Sentence

 ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),
           ref3~donkey~brown, ref1~man]
 ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]
 ** s:[beat(pres,pres,perf,act,indic),ref10~male!,ref8~person!] ▸
        [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]
 ** s:∀ind5.[give(pres,pres,perf,act,indic),
               ref13~donkey!,
               ref15~carrot×ind5,
               ref11~woman!]
 ** s:∀ind7.[eat(pres,pres,perf,act,indic),
               ref19~carrot!×ind7,
               ref16~donkey!]
 ** s:∀ind8.[ride(pres,pres,perf,act,indic),
               ref22~donkey!,
               ref21~person!×ind8]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey~male
ref7 is bound to e2~brown~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~brown~donkey~male
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot
ref21 is bound to e3~woman
               and e1~man~unkind
ref22 is bound to e2~male~donkey

>>
```

Figure 11: *Continued George output for "The people ride the donkey."*

Finally on the subject of unions of sets, note that, again, there is no difference in treatment between entity tokens specifying individuals and those specifying sets. Thus, we are still working with the very simple top-level algorithm presented at the beginning of this chapter, to which we have added nothing beyond the rule needed to restrict reference on grounds of number. The result of this is that George never does more work than is strictly necessary to analyse set reference, because sets are treated internally as individual objects (not unlike *arbitrary objects* – see [Fine, 1985]), and treated like any

other such, until there is no alternative to viewing them as agglomerates – maybe because sort information has been applied explicitly to some elements but not others. Dividing sets into agglomerates of subsets is the subject of the next section. Note that this "lazy" view of the language understanding operation is in perfect accord with the notion of adaptable representation and incremental parsing and reference evaluation, because the representation of sets is uncommitted as to the exact form of the application of any relations expressed in the discourse to the members of sets. It also agrees with [VanLehn, 1978]'s suggestion of postponed evaluation in humans.

6.5. Indexed Reference to Subsets

In referring to a subset of a set, there are broadly two possible referential behaviours. One is where a previously uniform set (represented in George by a single entity token) is divided into two by the discovery of new sort information about some (but not) all of its elements. The other is where a union of sets, represented by more than one token, like that discussed above, is broken again into subsets corresponding with the original tokens, which is equivalent to referring to (one of) the original separate referents, and thus does not present an interesting problem. In the former case, the information which selects a subset by excluding those elements not in the subset need not be complete – it is enough to know that they are not inconsistent with the negation of the new property. For example, in "I own four t-shirts; two are blue." the two de-selected items simply acquire the property [colour isnt blue]; we do not need to know any other t-shirt colours. If this information is inconsistent with existing sort information in the excluded set, then the members with the inconsistent properties should have been included in the selected subset in the first place. The intermediate behaviour, where a union of sets is divided in such a way that the set(s) specified by one or more of its existing entity tokens is(are) divided may be viewed as a generalisation of the first case, above. Examples of these three cases are given in 158. The explicit number information is added here to make the examples clearer – the same effect is achievable without it. In each case the subset reference is underlined.

"Five donkeys go on holiday. Three monkeys go with them.
The holiday-makers decide to fly to Majorca.
The ones with hooves have trouble getting on the plane." 158a

"Five donkeys go on holiday.
They set off for Majorca, but the two brown ones end up in Clacton." 158b

"Five donkeys go on holiday. Three monkeys go with them.
The holiday-makers decide to fly to Majorca.
The two fat animals have trouble getting on the plane." 158c

Here, then, we are faced with a situation not unlike one which has arisen previously. All the way through this discussion, we have allowed sort information to propagate from references along bindings and add to the information already included in entity tokens. Subset reference can be treated in exactly the same way, except that in cases b and c of the example above, one or more of the entity tokens involved must be divided into two new tokens, one having the property defining the subset and one having its negation. These will then specify the selected and de-selected subsets of the discourse entities specified by the original entity token, respectively. Once we have accepted George's view of binding to entity tokens rather than to explicit sets, such an approach is not unreasonable.

The obvious, brute force way to represent this in George would be by the creation of two new entities with all the properties of the old one, and with the addition of the new property, negated on one of them. This would represent explicitly the selected and de-selected subsets. The two new entity tokens could then be globally substituted for the old. This idea works well enough in principle, but can be extremely inefficient if the discourse is long – particularly if it involves many subset references, with which the number of entity tokens increases combinatorially.

A more elegant approach would be to include a tree of entity tokens in the Entity Tokens database from the point at which the set was divided, showing how the division occurred. Such a tree could represent the relationship between the "parent" entity token (at the root) and its more refined offspring (as leaf or branch nodes). Then, instead of an entity token being removed immediately from a candidate set when it was found to be inconsistent, we would first attempt to move "down" one of the branches of the tree to a related, more specific token. Only after a branch has been followed to its leaf would we be licensed to remove the entity token from the candidate set, by the need to move beyond that leaf to find a suitable refinement of the entity token. This idea has the advantage that information about the introduction of sets is preserved and easily available – it may be necessary later in the discourse to know that all the elements of a set were introduced at the same time, as in 159, in whose final sentence a (meta-)reference is made to the initial form of a set, before it was divided by subsequent references:

"I bought five carrots last week. Some of them were rotten; some were fresh.
I bought five more yesterday. I gave the first batch to Pedro the donkey." 159

This information is less easily available (or, indeed, it may be lost altogether) after the global replacement required by the brute force method.

The tree representation is also potentially substantially more efficient than the brute force approach suggested before. In particular, that approach requires that all occurences of the old entity token in the existing bindings be replaced with the new tokens. This may be a large task, especially if a set has been divided a large number of times – which is frequently the case in real discourse. The tree representation, however, requires no change to earlier bindings, because the relation between the old and the new entity tokens is explicitly maintained. Less importantly, subsequent references to the whole set are more efficient, because they are bound to one entity token and not to two (or more) with the corresponding overhead. Finally, the tree representation is easily built in to the George mechanism we have developed in this chapter, by a small addition to sub-process ③ in the top-level algorithm, rather than a major change to the top-level algorithm itself, as would be required by the brute force method.

I will refer subsequently to this divison of entities in trees as *entity token partition* and, unsurprisingly, to the tree produced as an *entity token partition tree*. The operation of entity token partition is very important in the process of reasoning about set reference explained in Chapter 7. It is defined formally in Rule 9.

Let E be an entity token bound to an indexed reference. Then:

Select new entity tokens E_1 ... E_n from Ent, apply to each the sort information applied to E, and insert them in the Entity Tokens Database. Add the pairs $\{\langle E, E_i\rangle \mid 1 \le i \le n\}$ to the partial order $\gg$.

The Entity Token Partition Tree of E is then defined as

$$\{E_i \mid 1 \le i \le n\} \cup \bigcup_{i \le n} T_i$$

where $E \gg E_i$ and T_i is the Entity Token Partition Tree of E_i.

Rule 9: *Entity Token Partition*

7. Summary

In this chapter I have presented the following material.

1. The top level algorithm of the George DeReferencer, with sub-algorithms covering
   Introductory simple reference;
   Non-introductory simple reference;
   Context extended reference.

2. The rules involved in
   Maintenance of referential consistency;
   Reference refinement;
   Context extension comparison and relevance assessment.

3. The emergent effect from the top-level algorithm of 1, above, of reassessment of a default non-introductory view of a reference to be introductory in response to failure of non-introductory reference.

4. The structural referential constraint rules and their maintenance, including
   The "closed non-coreference" rule.

5. The application (NB not extension) of the ideas in 1, 2 and 3, above, to
   Introductory indexed reference (to whole sets);
   Non-introductory indexed reference (to whole sets);
   Indexed reference to split sets;
   Indexed reference to subsets.

6. The addition to structural referential constraints in 4, above, of
   Number restrictions on reference.

The argument presented in this chapter includes a new treatment of noun-phrase (post-)modification, and an associated view of matching data appearing in noun-phrase post-modifiers which depends on, and therefore partly vindicates, the decision to keep separate George references and Entity Tokens. Apart from this new aspect, most of the work presented in this chapter has effectively been the casting of [Mellish, 1981]'s and [Webber, 1979]'s work into an adaptable system suitable for the kind of manipulation which I will present in Chapter 7, and showing that doing so need not compromise the system's ability to perform the tasks performed by those systems. This said, the presentation gives a rather different slant on the problem, in that it makes a virtue of expressing dereferencing behaviour in a number of simple rules, rather than in the single universal rule favoured by both Mellish and Webber.

8. Afterword

The ground work for presentation of George's approach to representing and reasoning about references to underspecified sets is now complete. In the next chapter, I will explain how George's use of entity tokens, bindings, and references in representation of discourse facilitates such reasoning, and how the adaptability of GRL and George's set representation makes a potential combinatorial explosion computationally tractable.

Chapter 7
Quantified Reference to Underspecified Sets

Abstract

The intuitions behind indexing of references are explained. The linguistic effects captured by the two different kinds of index operator are introduced.

Index propagation and expansion, allowing explicit representation of the application of predicates to elements of sets, are introduced, in conjunction with the notion of index dependency. These operations allow the readings of ambiguous but fully specified weak quantifiers to be enumerated.

The behaviour of underspecified set references is introduced, leading to the generalisation of the index expansion operation to index partition. The behaviour of strong and weak indices under index propagation is detailed.

Finally, index propagation is extended to deal with context extended reference, enabling George to deal with "donkey sentences" and other dependent reference.

1. Introduction

In this chapter, I will explain the main focus of the work in George, which is a means of representing and reasoning about sets which are underspecified in number. The explanation will follow on from that in the last chapter – the representation and general philosophy used will be the same.

Before I attempt to explain the detail of representing underspecified reference in the George style, it will be useful to discuss the intuitive ideas behind the representation, which we can do with reference to examples presented in Chapters 1 and 4.

The work presented in this chapter is an example of how adaptable representations can be used to facilitate automatic language analysis in situations, frequent in real discourse, where ambiguity or vagueness is potentially so great that search through an immediately enumerated space of possible readings is not in general computationally viable. Thus, while the technique is presented as interesting in itself, the overall concept of adaptability, of which number-underspecified set reference is just one aspect, is still the primary underlying theme.

2. Representation of Sets by Indexing

First, in explaining George's view of sets and underspecified reference to them, I must reiterate exactly what I mean by underspecified set. The research presented here focusses on set references which are in a sense ambiguous because they refer to sets whose cardinality is unknown but whose members have known properties. I use the term *underspecified* to describe such references, because there is a distinct intuitive difference between this ambiguity or vagueness, where we know that our referent is a subset of some definite known homogenous set of undifferentiable entities (*eg* "Half of them" in 160), and the kind more often discussed, where we are attempting to choose from a number of explicitly differentiable entities (*eg* "The fat one" in 161) or translations. This distinction blurs when we have a set of distinct entities and we attempt to refer to some subset of them; however, George can deal with this case too.

"Some men were in the room. Half of them owned donkeys." 160

"Two thin men and one fat man were in the room. The fat one owned a donkey." 161

I mentioned in Chapter 1 that I am not concerned here with collective nouns, but only with sets which are referred to by genuinely plural references. Dealing with collectives (see Chapter 9) is a further level of linguistic abstraction, as it were, which would only obscure the central theme of this research.

To start with, then, we need a general notation for making predications of elements of sets, and, in particular, one which will allow us, should the need arise, to sub-divide the sets – explicitly – and make further predications on the resulting subsets and their members. Consider, again, the example given in Chapter 4, adapted from [Webber, 1979]:

"Jim bought two t-shirts. 162a

The blue one was faulty." 162b

I pointed out before that, since the set of entities is small, we could represent it something like this (ignoring tense and reading $\oplus$ as "exclusive or"):

"bought(t-shirt1, Jim) $\wedge$ bought(t-shirt2, Jim) $\wedge$

(blue(t-shirt1) $\wedge$ faulty(t-shirt1) $\oplus$ blue(t-shirt2) $\wedge$ faulty(t-shirt2))" 163

However, especially if we want to represent large sets, we risk a combinatorial explosion if we try to use this rather simplistic approach. More interestingly, if the cardinality of the set is not defined at all, as in

"Jim bought some t-shirts. 164a

The blue ones were faulty." 164b

this solution is even less helpful, since we cannot make an exhaustive list of possibilities.

As I explained in the last chapter, this kind of "incomplete" reference is made available for reasoning in George by separating the number information in the reference(s) defining the set from the corresponding sort information. We view the latter as being part of a kind of arbitrary object or prototype which, in conjunction with the number information, in some sense *specifies* what is the set of discourse entities to which the reference(s) refer. By doing so, we are able to talk about the properties of the entities in a set without consideration of the number information available about the set – which seems, at least on a naïve intuitive level, to be exactly what happens when we listen to a discourse including such information.

The translations into GRL, then, of these partial discourses are as follows.

162:	∀ind1<2.[bought, ref2~t-shirt×ind1, ref1~Jim!]	165a
	coref(ref4~faulty, ref3~blue)	165b
164:	∀ind1.[bought, ref2~t-shirt×ind1, ref1~Jim!]	166a
	∀ind2.coref(ref4~faulty, ref3~blue!×ind2)	166b

(Remember that GRL does not directly represent the association between the entities of the first and of the second sentence; this side of things is handled entirely by the DeReferencer.)

Now, consider again example 164. When we have translated 164a to give 166a, we may draw a diagram like Figure 12a to represent the bindings and entity tokens. The arrows in Figures 12 represent bindings between references and sets of entity tokens, and may be read as "refers to".

The quantifier is viewed as part of any reference with which its bound index is associated by the index operator, and so the information it carries must appear in the binding. In Figures 12, where the reference part of the bindings is shown in the boxes at the blunt end of the arrows, quantification is represented by the application of the index operator and an

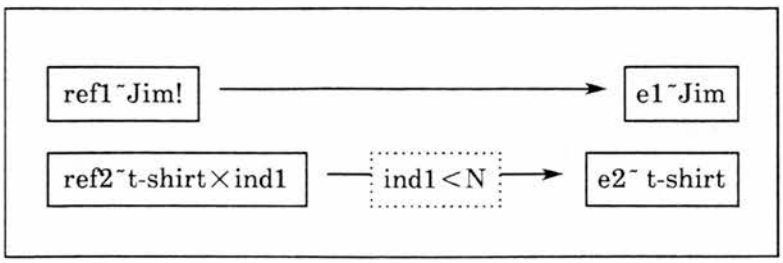


Figure 12a: Entities and Bindings for Example 164a/166a

index. The information conveyed by the upper bound of the binding is then shown in the box attached directly to the arrow. The set of entity tokens at the other end of the binding is shown as one or more boxes at the sharp end of the arrow.

In the case of example 164a/166a, the upper bound on the quantification is unspecified. Therefore, the upper bound is written (in the diagrams) simply as N or M, representing some uninstantiated number in ℕ.

The attachment of a number, N, to the arrows may be thought of as shorthand for N bindings between N distinct but characteristically identical references to N distinct copies of the entity token. In a case such as this where N is unspecified, it is impossible to draw the complete expansion.

Now, consider what happens when we translate 164b to give 166b. After reading the first noun phrase, we need to subdivide our set of t-shirts into those which are blue and those which are not. We can represent this first stage with a diagram like Figure 12b, where M

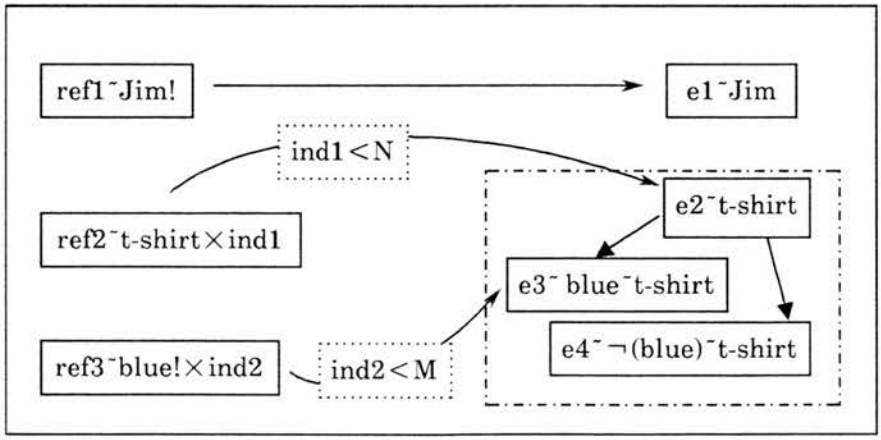


Figure 12b: Entities and Bindings for Example 164b/166b

is the (unspecified) number of blue t-shirts. Note that the negated sort in the diagram is

only a shorthand form, and would in fact be represented by a sort whose defining set contained just the property value pair

[colour isnt blue]

rather than by a declarative application of a negation operator to a sort.

Finally, we are able to introduce the information from the coref predicate in 166b – namely that the sort applied to ref4 applies to the entity(ies) referred to by ref3 – to produce a diagram like Figure 12c. In this fairly simple case, then, we have a means of

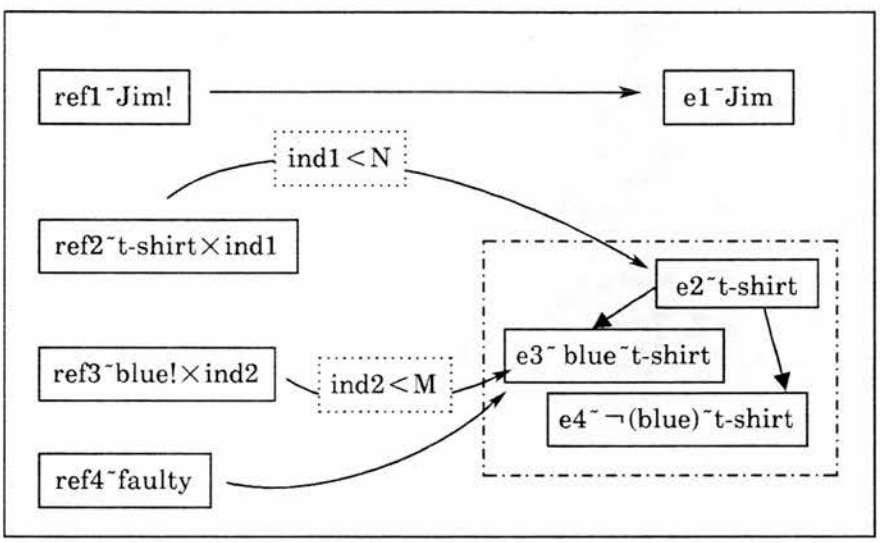


Figure 12c: Entities and Bindings for Example 164b/166b

computing and representing set reference.

3. Using Indices to Reason about Quantification

3.1. Introduction

The examples in the last section show the simple intuition behind George's representation of sets: that any quantified reference is a compact representation for some larger number of references, and hence that a GRL expression containing such a reference is a compact representation for some larger number of expressions. In particular, that larger number may be unknown.

However, none of the worked examples presented so far have involved references to subsets, undistinguished by sort, of homogenous sets. This makes reference particularly

easy to analyse, as I showed in Chapter 6. Now we must consider what happens when this more complex kind of reference occurs in an utterance.

Before starting this discussion, it is important to understand that the processes described in the following sections are not gratuitously applied to sentences in George – that is, they only apply when there is a reason for them to do so; when, for example, a reference is made to an individual introduced as one of an homogenous set. In this event, George's Discourse Memory may be altered to account for any deductions which can be made from the later reference regarding the nature of that set. Thus, search through the very large space of possibilities I will consider here is always guided, in the weak sense that the inference process always has a target. Adaptability of the representation of those parts of entities unaffected by the inference is not compromised. Therefore, as elsewhere, non-determinism is kept to a minimum.

In particular, the reasoning which I will describe in the following sections takes place as a sub-process of the overall dereferencing algorithm specified in Chapter 6: the algorithm is reproduced below; the sub-process described in this chapter is part 3.

> On receipt of a set of references from the Parser, for each reference symbol in the set, perform each one of the following actions (the labelling is for convenience later).
>
> If the reference symbol is new to the DeReferencer① then
>
> 1. a. create a new binding for it, then...
>
> b. If its reference is definite②, find the names of the entity token(s) to which it may refer③ and insert them into the binding;
>
> c. If its reference is indefinite②, create a new entity token and insert it into the binding;
>
> otherwise
>
> 2. a. look up its existing binding, then...
>
> b. Combine any new information④ about the reference with that already contained in the binding, discarding any erstwhile candidate entity tokens which are now made inconsistent③. Check that entity tokens removed are not left newly unbound; if any are, remove them from the Entity Tokens database.
>
> 3. If the binding of this reference contains no entity tokens (*ie* if the reference is completely new, or no existing token was found to be consistent, or all the tokens previously found to be consistent were deleted on grounds of inconsistency on this latest iteration), search backwards through discourse memory for expressions

containing references bound to entity tokens of consistent sort with this reference. Apply the underspecified reference mechanism[6] to each such expression until one is found which yields success under that mechanism; then rewrite the bindings and if necessary the expression itself to create entity tokens for the bindings in the current sentence.

Finally,

4. If the reference is definite and its binding contains no entity tokens (*ie* if all the tokens previously found to be consistent were deleted on grounds of inconsistency on this latest iteration), create a new entity token with the appropriate properties[5] and insert it into both the Entity Tokens database and the binding.

In the following sections, I will describe the processes involved in reasoning about set reference referred to in part 3. In particular, I will suggest a means of deducing from an ambiguous quantified expression a conjunction of expressions with a subset of the same semantic interpretations. This will allow us to represent sets in the simple and tractable way outlined before, but still deal with complicated situations where the exact detail of the application of a predicate to the members of its domain is not specified. These processes, therefore, have a status roughly equivalent to that of refinement of a set of candidate entities in the easier referential cases explained before; they form part of the process of inference about descriptions of entities in the discourse world. As such, they are not unlike Webber's attempts [Webber, 1979] to discover what is "available for reference" in a discourse.

However, it seems that this harder kind of set reference takes second place to more straightforward readings. That is to say, in the event of ambiguity, a reading of a sentence not involving the breakdown of existing sets will be chosen before one requiring such a breakdown. In the last chapter, I explained that, for simple references, if we fail to find an entity token suitable for binding (under the criteria given before) then we simply create one. The process described in this chapter fits, in the George system, between such more straightforward analysis of indexed reference (which I explained in Chapter 6) and the decision to create a new entity token for a binding. Thus, the Principle of Parsimony is upheld.

Once a reading has been deduced which fits the requirements imposed by the discourse, we may partition entity tokens in our database, as explained before, to represent the new (sub)sets created, and we may replace the original quantified expression in the Discourse Memory with a new version, which may be differently quantified or not quantified at all,

depending on the exact circumstances at the time. These details will be covered later in this chapter.

3.2. Strong and Weak Quantification

Before proceeding with the examples and discussion of the effects of quantification covered in George, I must introduce a new notion into the linguistic framework. This is the distinction between strong and weak quantification.

I have already introduced operators to denote strong and weak indexing, but without giving a motivation for doing so. The motivation, then, is that the two operators give us an explicit way of representing the two forms of quantification, in line with the general policy here of representing surface form as explicitly as possible. So what differing surface forms we are representing?

In [Mellish, 1985], Mellish points out the need for different respective treatments of ordinary plural noun phrases, and what he calls "*each* phrases"; such a distinction is also implicit in the results of VanLehn's survey of rules for interpreting quantification in natural language [VanLehn, 1978]. (This analysis does not include the classical existential quantifier, whose semantics is rather different.)

The point is that different quantifiers allow us to express differences in the way the detail of such quantification is worked out. For example, consider two simple discourses like those shown in 167a and b.

"The men own a donkey." 167a

"Each man owns a donkey." 167b

These two have in common a set of men of arbitrary (possibly predetermined) cardinality, a (possibly singleton) set of donkeys, and a relation between the two. In a, there is only one reading, where there is only one donkey which is owned by all the men. In b, the most easily accessible reading is where the mapping between the two sets is one-to-one – that is to say, every man in the set of men owns a different one of the set of donkeys. There are, however, a large number of other possible readings where each man owns some donkey, but the relation is not one-to-one, because some donkeys are owned by more than one man. One example of such a reading is that where there is exactly one donkey and every man owns it. The difference, in abstract, lies in the cardinality of the set of donkeys, and in the exact nature of the realtionship between elements of that set and the set of men. In the first sentence, there is exactly one donkey; in the second, there is some number of them –

from one up to the same number as of men. This is particularly significant in view of the fact that the surface form of the reference to the donkey(s) is the same in both sentences. Further, 167a may be read collectively or distributively, while 167b does not admit the collective reading.

I suggest that either or both of these effects may be meaningfully associated with the notion of "strength" of quantification, so that "each" is a strong quantifier and "the" is a weaker one. As such, it is appropriate within the George framework to use different notations for the translations of the two words, just as we used different representations for definite and indefinite references. (Indeed, one could go further, and say that we need a real sliding scale of quantifier strength, to account for these effects in full. For the purposes of experiment, however, it will be better to constrain the domain to weak and strong quantifications, and establish a general principle. Then, the idea can be later extended by closer study of the minutiæ involved.)

In order to denote these two strengths of quantifier, then, we need two symbols. It will become clear from the explanation in the following sections that a good way to express the behaviour we wish to discuss is not to quantify over sets of entities, but to view quantification as something applied to references. Thus, we can use one basic notion of quantification – applying predicates to the elements of sets, according with the linguistic intuition expressed earlier – and change the way it is applied to express the difference between strong and weak quantification. This, then, is why George uses indices instead of conventional quantifiers and their associated domains and why the application of the indices to references is denoted by two related but distinct operators, the strong and weak index application operators, $\otimes$ and $\times$, respectively.

Finally, note that strongly indexed references (which represent strongly quantified referring expressions) may only be definite – the non-definite reading of, say, "every" can only be read as generic in the case where it does not refer to an existing set. The connection between definites and indefinites and strong and weak indices seems to run deeper than this, as will be seen later; however, it is not clear what is the exact philosophical nature of the relationship.

In the following sections, I will explain how we can use this distinction to build a generalisation of the behaviour of quantified references with respect to each other and to other references. I will propose a mechanism which will enable us to produce the various explicit readings available from the weak quantifiers by a mechanical process.

I will start by considering the various combinations of quantifier in sentences containing only two-place predicates, and then show that it extends to N-place relations. Finally, I will extend the idea to cover dependent references – that is, references related to other quantified references by subordination.

4. Restrictions on Quantified Reference

It is important to understand here that the expressions generated by the rules I will formulate here are, in the same way as [Webber, 1979]'s discourse entities, what is available for reference. In particular, the examples will contain relations, such as ownership, which do not have any particularly strong pragmatic connotations. Some of the readings which will be generated could, if they contained relations like "give", for example, lead to pragmatic contradictions. For example, the sentence in 168a can only be read meaningfully if each of the implicit giving actions takes place at a different time.

"Every man gives a donkey the carrot." 168a

"The men give a donkey a carrot." 168b

Such pragmatic effects are related to the strength of the quantification – the same problem does not arise with 168b, because of the collective nature of the sentence, reflected in the weakness of the determiner, "The". Nevertheless, it should be noted that the need to rule out such spurious readings does not detract from the utility of the methods presented here; indeed, such analysis might best be characterised as a process based on the output of those methods. As such, it is outside the scope of this discussion.

5. Sentences containing more than one Quantifier

In the last chapter, I discussed the behaviour of the George system in response to a reference to a subset of a set previously introduced into the discourse. Such reference was relatively uncomplicated, because the sets in question always appeared in relations with individual objects, and because I did not discuss the exact form of the quantification. Thus, the exact nature of the relation between the members of the set was never ambiguous.

In this and the following sections, I explain how George deals with the harder cases, where relations are specified between two or more non-singleton sets of entities. In general, linguistic specifications of such relations do not contain all the information needed to give their full logical or mathematical detail. George's main feature is its ability to propose

readings of such quantified sentences, so that subsequent references to the entities involved may guide inference of the (more) exact nature of the relationship expressed by the original sentence.

This process is characterised in George as a search through a space of possibilities. As specified in the algorithm given earlier, the search only begins when the other, more constrained methods of finding entity tokens to which references may be bound have failed. If we do have to call on this search, the sequences of search steps (which correspond with sequences of inference steps in reasoning about the nature of set references) always terminate, either when an appropriate token has been found, or when no further inference steps can be applied. Search through this space of possibilities is non-deterministic. In the event that a chosen analysis is proved incorrect later in the discourse, another choice is sought which fits both the original constraint(s) and the new one(s) which rendered the former analysis inadequate. As I mentioned before, maximal possible adaptibility is maintained during the inference process, particularly of those parts of the discourse representation not directly affected by it. Thus, resatisfaction of a search may not involve actual backtracking in the system.

It will become clear in the examples later that this search can be guided to a large extent by features of the existing discourse memory. However, such guidance has been left here to an oracle. For the purposes of experiment, blind search is adequate, since it is not the speed with which we generate our resulting expressions which is of interest here, but rather the generation itself. Thus, pointers given through the following text as to why an inference step is well-chosen are given with the benefit of higher level knowledge. George cannot currently use them.

6. Weak Indexing

6.1. Introduction

Consider the discourse shown in example 169. This is the first sentence of the puzzle in example 1, which will constitute the main running example of this chapter. It contains only weak quantifiers.

"Some men own some donkeys." 169a

∀ind1.∀ind2.[owns,ref2˜donkey×ind2,ref1˜man×ind1] 169b

As before, if we had upper bounds on the indices, we could write out an exhaustive list of sentences representing the meaning of the utterance. However – and this the whole point here – there are many ways of doing so. Equally, in unbounded cases like 169, there is a potentially large number of interpretations, each of which is impossible to write exhaustively. Consider, for example, the following situations.

1. Each man owns exactly one donkey, and each donkey is owned by exactly one man.

2. Each man owns exactly one donkey, and at least one donkey is owned by more than one man.

3. Each man owns a non-empty set of donkeys, and no donkey is owned by more than one man.

4. Each man owns a non-empty set of donkeys, and any donkey may be a member of more than one such set.

5. Each donkey is owned by a set of men, and the intersections between the sets are empty.

And so on...

This list is by no means complete, so if we are to find a general method of reasoning about such statements, we need a very general representation and inference system. I will therefore first present a naïve solution to a rather easier, more specific example – that shown in 170 – where we can (NB for the purposes of argument only) enumerate the various possibilities. I will then extend this to the greater generality of fully specified reference, and thence, later, to the harder underspecified references shown above.

"Two men own two donkeys." 170a

$\forall ind1 < 2.\forall ind2 < 2.[own,ref2\check{}donkey \times ind2,ref1\check{}man \times ind1]$ 170b

The five possible readings are shown diagrammatically in Figures 13a-e (spread throughout the exposition), the ownership relation being indicated by the arrows. I will discuss the possible readings and George's mechanism for obtaining them one by one.

Initially, to represent the five readings directly in GRL, we use the expression shown in example 170b. As we have seen, this ambiguous representation is adequate for referential purposes, until we need to talk about individuals or subsets. For the moment, I will

restrict our discussion to individuals (which are a special kind of subset – *viz* a singleton subset).

The process I am about to describe can be divided into four sections, as follows. Its application is caused by the failure of an indexed reference to refer successfully in number to an entity to which it may refer successfully by sort, as defined in part 3 of the top level algorithm.

1. Optional application of *Index Propagation* – manipulation of the application of quantification to the various references in an expression. This gives the same effect as changing quantifier scope in more conventional notations.

2. Optional application of *Index Dependency* – linking of two indices together to indicate that the sets over which their quantifiers range are in one-to-one correspondence with respect to the predicate in the expression. This operation is analogous to [Mellish, 1981]'s "linked dependencies".

3. Optional application of *Index Expansion* or *Index Partition* – manipulation of ambiguous references to sets to yield more specific (conjunctions of) references to (sub)sets.

4. If Index Expansion was applied in 3, *Reference Renaming* – application of a function mapping Ref×ℕ→Ref (as defined in Chapter 4) such that the output of the function is a unique reference symbol. This simply ensures that different references in the system after index expansion do not have the same name.

6.2. The Cross Product Reading: Weak Index Expansion.

The Cross Product reading of sentence 170 is that represented in Figure 13a, where each of the two men owns both of the two donkeys. It is easily represented using conventional logical (FOPC) notation, as shown in example 171a. 171b shows an alternative, expanded reading, which might be more suited for subsequent reference analysis.

$$\forall x.(\, donkey(\, x\,) \Rightarrow \forall y.(\, man(\, y\,) \Rightarrow owns(\, x, y\,)\,)) \qquad 171a$$

$$\exists a.(donkey(\, a\,) \wedge \exists b.(donkey(\, b\,) \wedge \exists c.(man(\, c\,) \wedge \exists d.(man(\, d\,) \wedge owns(\, a, c\,) \wedge owns(\, a, d\,) \wedge owns(\, b, c\,) \wedge owns(\, b, d\,)))))) \qquad 171b$$

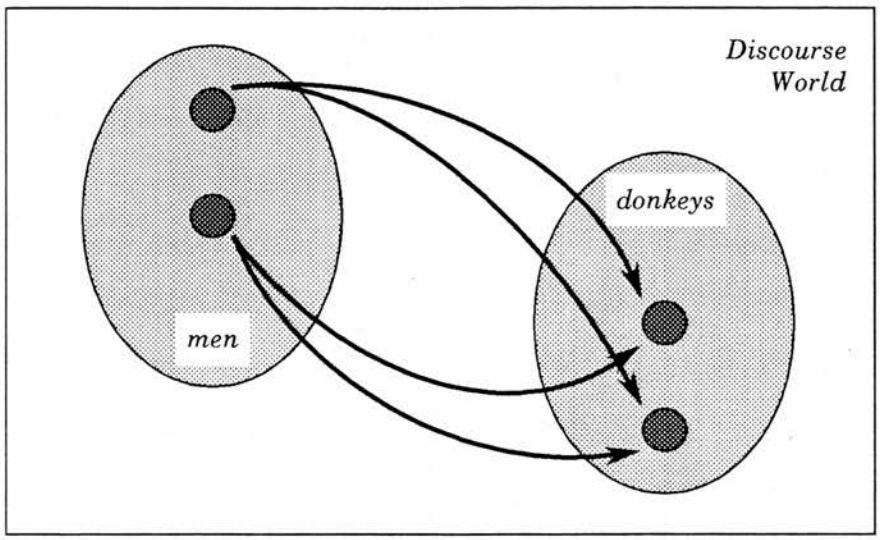


Figure 13a: *Each one of two men owns both of two donkeys.*

To produce an explicit representation of this reading (where the quantifiers are expanded into a conjunction of unquantified expressions like 171b), George uses *Weak Index Expansion*. In Chapter 4, I defined indices as being variables over $\mathbb{N}$ or some contiguous subset of $\mathbb{N}$ including 0. In this example, we have upper bounds on both of the indices: they both range over $\{0, 1\}$. We would like a means of expanding the indices in such an expression to give an equivalent unquantified set of expressions, replacing each index in a sentence with each of its possible values. This would give us a more direct representation of the relationship between the elements of the two sets.

Weak Index Expansion is defined in Rule 10. Example 172 shows the operation at work, first expanding the indices (172b) and renaming the resultant references (172c, which also shows the resulting entity token partition trees). Note that only steps 3 and 4 of the four steps specified in Section 6.1 are being applied here.

$$\forall ind1 < 2.\forall ind2 < 2.[own,ref2\text{~}donkey \times ind2,ref1\text{~}man \times ind1]$$

$$\{\langle\{\forall ind1 < 2.ref1 \times ind1\},\{e1\}\rangle,\langle\{\forall ind2 < 2.ref2 \times ind2\},\{e2\}\rangle\} \qquad 172a$$

$$[owns,ref2\text{~}donkey \times 0,ref1\text{~}man \times 0]$$
$$[owns,ref2\text{~}donkey \times 1,ref1\text{~}man \times 0]$$
$$[owns,ref2\text{~}donkey \times 0,ref1\text{~}man \times 1]$$
$$[owns,ref2\text{~}donkey \times 1,ref1\text{~}man \times 1] \qquad 172b$$

$$[owns,ref5\text{~}donkey,ref3\text{~}man]$$
$$[owns,ref6\text{~}donkey,ref3\text{~}man]$$
$$[owns,ref5\text{~}donkey,ref4\text{~}man]$$

Let $I_1...I_p$, $J_1...J_q$ be index symbols, I be a bounded index symbol with upper bound N, and K be an arbitrary, possibly empty, string of I_i and J_i conjoined by $\times$ and $\otimes$. Let R, $R_1...R_n$, $S_1...S_m$ be references (n, m, p, q, $N \in \mathbb{N}$), and P be a predicate symbol. Then:

We can rewrite an expression of the form

$$\forall I_1...\forall I_p.\forall I<N.\forall J_1...\forall J_q.[P,R_n...R_1,R\times I\times K,S_m...S_1]$$

by creating N of copies of the expression each with I replaced by a different member of $\mathbb{N}$ less than N, iff N is instantiated with a known value, thus:

$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R\times 0\times K,S_m...S_1]$$
$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R\times 1\times K,S_m...S_1]$$
$$\ldots$$
$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R\times (N\text{-}1)\times K,S_m...S_1]$$

After rewriting, each reference symbol with an integer applied to it is replaced by a new unique reference symbol; all occurrences of each ⟨reference symbol,integer⟩ pair are replaced by the same symbol. Weak index expansion causes a corresponding entity token partition (if one does not already exist), so each new reference has its own distinct entity token. Having done so, bindings containing the expanded reference must be replaced by n bindings between the new references and entity tokens.

Rule 10: *Weak Index Expansion*

$$[\text{owns,ref6}^{\smile}\text{donkey,ref4}^{\smile}\text{man}]$$
$$\{\langle\{\text{ref3}\},\{\text{e3}\}\rangle,\langle\{\text{ref4}\},\{\text{e4}\}\rangle,\langle\{\text{ref5}\},\{\text{e5}\}\rangle,\langle\{\text{ref6}\},\{\text{e6}\}\rangle\}$$
$$\{\text{e1} \gg \text{e3}, \text{e1} \gg \text{e4}, \text{e2} \gg \text{e5}, \text{e2} \gg \text{e6}\} \qquad 172\text{c}$$

In a fuller implementation of GRL, the four resulting sentences might be written conjoined, to express their common origin. Here, though, in the context of the GRL semantics supplied in Chapter 4, they must be written separately, because conjunction of closed predicates is not defined. This makes little difference here, because there is an equivalent notion in the semantic translation algorithm – the Discourse Memory is viewed as one big conjunction.

Note that the result of the expansion represents the semantics we require only if a corresponding entity token partition is performed, as specified in the rule, because

otherwise, the new references would simply refer to the same set as before, which would be incorrect.

6.3. Weak Index Propagation

All other readings of sentence 170 involve more complicated interaction between the quantifiers in the sentence. In many systems, this is represented by the application of different quantifier scopes, affecting the syntactic structure of the sentence. In George, the interaction is viewed as a propagation of the effect of one quantifier to another through, as it were, the predicate connecting them. This is represented in GRL through the *Weak Index Propagation* operation, specified in Rule 11. (The definition includes a reference to

Let I be an index symbol, R, $R_1...R_n$, $S_1...S_m$ be references ($n, m \in \mathbb{N}$), and P be a predicate symbol. Then:

Given an expression of the form

$$\forall I.[P,R_n,...,R_1,R \times I,S_m,...,S_1],$$

we can replace any number of R_i with $R_i \times I$ iff

1) **the index I only appears once within this closed predicate (ie it has never been propagated);**
2) **the Index Application Rule (Rule 18) holds for $\langle left,R_i,R \times I\rangle$.**

To maintain consistency, I must also be propagated to any identical occurrences of R_i in other expressions in the discourse memory. If appropriate, quantifiers should be added.

Rule 11: *Weak Index Propagation*

Rule 18, the *Index Application Rule* of Section 16, which specifies the kinds of references to which indices can propagate. Here, that rule is always satisfied, because we are only moving weak indices to weak indexed indefinite references, which is always an acceptable combination.)

Weak Index Propagation allows weak indices to move around the arguments in the closed predicate in which they appear. Note that the operation is only defined on references whose index is unique. This means that an index cannot propagate from a reference on to which it has been propagated. Propagation allows an index to be copied, leftwards, onto weak indexed references other than the one with which it is explicitly associated by the

input, but within the same closed predicate. I will say that the newly arrived index then *dominates* the older one (as in FOPC some quantifers dominate others).

It is important to note also that it is not possible to read sentence 170 in such a way that two sets of two men own one each of a pair of donkeys. Therefore, weak index propagation operation is only defined from right to left (*ie* from subject to object, and to indirect object, if applicable) and not in the reverse direction.

6.4. The One-to-Many Reading: Weak Index Propagation

One of these more complicated readings of 170 is that illustrated by Figure 13b. This is

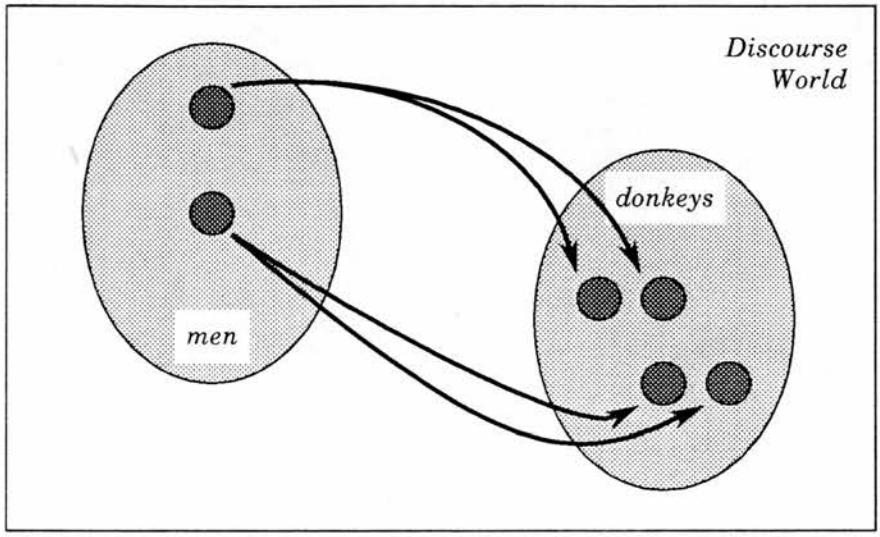


Figure 13b: *Each of two men owns both members of one of two non-intersecting sets of donkeys.*

the reading where each man owns a distinct set of two donkeys. In George, the reading is obtained by applying weak index propagation, and then weak index expansion, as shown step by step in example 173. (In an attempt to make the examples in this chapter more readable, I have omitted sort information from the bindings shown. This is merely for the purposes of example, like the simplified notation used for predicate names here.) In association with the corresponding entity token partition, as required by Rule 10, the resulting conjunction represents the desired reading.

$$\forall ind1<2.\forall ind2<2.[own,ref2^{-}donkey\times ind2,ref1^{-}man\times ind1]$$
$$\{\langle\{\forall ind1<2.ref1\times ind1\},\{e1\}\rangle,\langle\{\forall ind2<2.ref2\times ind2\},\{e2\}\rangle\} \qquad 173a$$
$$\text{—index propagation}\rightarrow$$

$$\forall ind1<2.\forall ind2<2.[own,ref2^{\smile}donkey\times ind2\times ind1,ref1^{\smile}man\times ind1]$$
$$\{\langle\{\forall ind1<2.ref1\times ind1\},\{e1\}\rangle,$$
$$\langle\{\forall ind1<2.\forall ind2<2.ref2\times ind2\times ind1\},\{e2\}\rangle\} \qquad 173b$$
$$\xrightarrow{\text{index expansion}}$$
$$\forall ind2<2.[owns,ref2^{\smile}donkey\times ind2\times 0,ref1^{\smile}man\times 0]$$
$$\forall ind2<2.[owns,ref2^{\smile}donkey\times ind2\times 1,ref1^{\smile}man\times 1]$$
$$\{\langle\{ref1\times 0\},\{e3\}\rangle,\langle\{\forall ind2<2.ref2\times ind2\times 0\},\{e5\}\rangle,$$
$$\langle\{ref1\times 1\},\{e4\}\rangle,\langle\{\forall ind2<2.ref2\times ind2\times 1\},\{e6\}\rangle\}$$
$$\{e1\gg e3,e1\gg e4,e2\gg e5,e2\gg e6\} \qquad 173c$$
$$\xrightarrow{\text{index expansion}}$$
$$[owns,ref2^{\smile}donkey\times 0\times 0,ref1^{\smile}man\times 0] \quad [owns,ref2^{\smile}donkey\times 1\times 0,ref1^{\smile}man\times 0]$$
$$[owns,ref2^{\smile}donkey\times 0\times 1,ref1^{\smile}man\times 1] \quad [owns,ref2^{\smile}donkey\times 1\times 1,ref1^{\smile}man\times 1]$$
$$\{\langle\{ref1\times 0\},\{e3\}\rangle,\langle\{ref2\times 0\times 0\},\{e7\}\rangle,\langle\{ref1\times 1\},\{e4\}\rangle,$$
$$\langle\{ref2\times 0\times 1\},\{e8\}\rangle,\langle\{ref2\times 1\times 0\},\{e9\}\rangle,\langle\{ref2\times 1\times 1\},\{e10\}\rangle\}$$
$$\{e1\gg e3,e1\gg e4,e2\gg e5,e2\gg e6,e5\gg e7,e5\gg e8,e6\gg e9,e6\gg e10\} \qquad 173d$$
$$\xrightarrow{\text{renaming}}$$
$$[owns,ref4^{\smile}donkey\times 0,ref2^{\smile}man] \quad [owns,ref5^{\smile}donkey\times 0,ref2^{\smile}man]$$
$$[owns,ref4^{\smile}donkey\times 1,ref3^{\smile}man] \quad [owns,ref5^{\smile}donkey\times 1,ref3^{\smile}man]$$
$$\{\langle\{ref2\},\{e3\}\rangle,\langle\{ref4\times 0\},\{e7\}\rangle,\langle\{ref3\},\{e4\}\rangle,$$
$$\langle\{ref4\times 1\},\{e8\}\rangle,\langle\{ref5\times 0\},\{e9\}\rangle,\langle\{ref5\times 1\},\{e10\}\rangle\}$$
$$\{e1\gg e3,e1\gg e4,e2\gg e5,e2\gg e6,e5\gg e7,e5\gg e8,e6\gg e9,e6\gg e10\} \qquad 173e$$
$$\xrightarrow{\text{renaming}}$$
$$[owns,ref6^{\smile}donkey,ref2^{\smile}man] \quad [owns,ref7^{\smile}donkey,ref2^{\smile}man]$$
$$[owns,ref8^{\smile}donkey,ref3^{\smile}man] \quad [owns,ref9^{\smile}donkey,ref3^{\smile}man]$$
$$\{\langle\{ref2\},\{e3\}\rangle,\langle\{ref6\},\{e7\}\rangle,\langle\{ref3\},\{e4\}\rangle,$$
$$\langle\{ref8\},\{e8\}\rangle,\langle\{ref7\},\{e9\}\rangle,\langle\{ref9\},\{e10\}\rangle\}$$
$$\{e1\gg e3,e1\gg e4,e2\gg e5,e2\gg e6,e5\gg e7,e5\gg e8,e6\gg e9,e6\gg e10\} \qquad 173f$$

6.5. The One-to-One Reading: Index Dependency

In the foregoing example, index propagation expressed the effect of one quantifier on another (which we traditionally get from scoping in, say, FOPC); and expanding the indices across $\mathbb{N}$ enumerated the elements of the sets. So far, the effect we have seen has been unsubtle: it might, for example, be expressed just as well through use of conventional universal quantifiers in FOPC. Now, we must consider more complicated relationships between the quantified sets.

The next reading we need to produce is that shown in Figure 13c. This is the reading

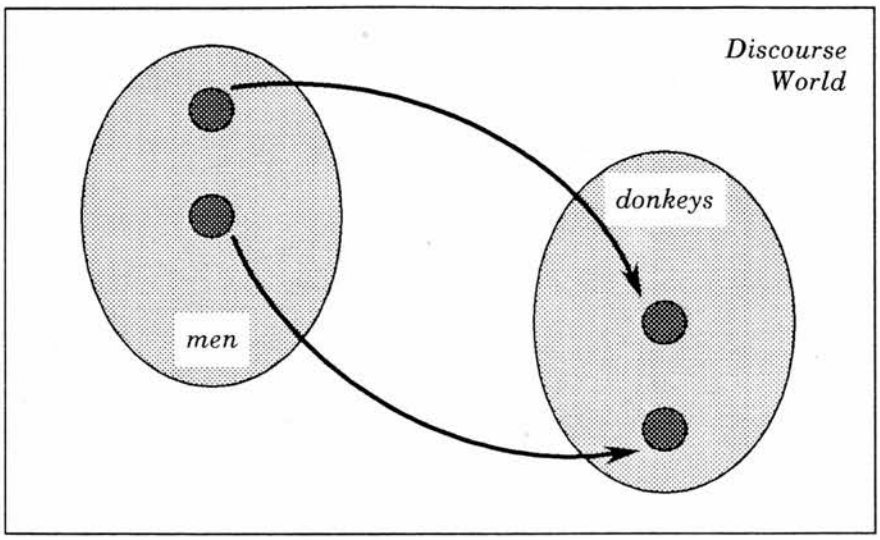


Figure 13c: *Each one of two men owns exactly one of two donkeys.*

where the sets are in one-to-one correspondence, so each man owns exactly one distinct donkey. To produce this reading, we use the notion of *index dependency*. The specification of the index dependency transformation given here is not complete – I will generalise it later (in Rule 12): for now this definition will suffice.

$$\forall \mathrm{ind}N_1.\forall \mathrm{ind}N_2.\mathrm{ref}N_3 \times \mathrm{ind}N_1 \times \mathrm{ind}N_2 \xrightarrow{\text{index dependency}} \forall \mathrm{ind}N_2.\mathrm{ref}N_3 \times \mathrm{ind}N_2$$

Note how the index and quantifier associated with the dominated index disappears: the dominating index has, as it were, gained complete control of the reference; this is like the use of a one-to-one correspondent skolem function in FOPC. The operation is equivalent in the George formalism to the use of linked dependencies in Mellish's MECHO.

The derivation of this reading is shown step by step in 174. The original sentence is shown in a. After applying index propagation (b), then index dependency to the reference with two indices (c), then applying index expansion (d) and renaming, we get the reading (e), which corresponds with a universal quantifier dominating an existential in FOPC, or with the use of one-to-one correspondent skolem functions.

$$\forall \mathrm{ind}1 < 2.\forall \mathrm{ind}2 < 2.[\mathrm{own}, \mathrm{ref}2\ \mathrm{donkey} \times \mathrm{ind}2, \mathrm{ref}1\ \mathrm{man} \times \mathrm{ind}1]$$
$$\{\langle \{\forall \mathrm{ind}1 < 2.\mathrm{ref}1 \times \mathrm{ind}1\}, \{e1\}\rangle, \langle \{\forall \mathrm{ind}2 < 2.\mathrm{ref}2 \times \mathrm{ind}2\}, \{e2\}\rangle\} \qquad 174a$$
$$\xrightarrow{\text{index propagation}}$$
$$\forall \mathrm{ind}1 < 2.\forall \mathrm{ind}2 < 2.[\mathrm{own}, \mathrm{ref}2\ \mathrm{donkey} \times \mathrm{ind}2 \times \mathrm{ind}1, \mathrm{ref}1\ \mathrm{man} \times \mathrm{ind}1]$$
$$\{\langle \{\forall \mathrm{ind}1 < 2.\mathrm{ref}1 \times \mathrm{ind}1\}, \{e1\}\rangle,$$

$$\langle\{\forall ind1<2.\forall ind2<2.ref2\times ind2\times ind1\},\{e2\}\rangle\} \qquad 174b$$

$$\xrightarrow{\text{index dependency}}$$

$$\forall ind1<2.[own,ref2^{\smile}donkey\times ind1,ref1^{\smile}man\times ind1]$$
$$\{\langle\{\forall ind1<2.ref1\times ind1\},\{e1\}\rangle,\langle\{\forall ind1<2.ref2\times ind1\},\{e2\}\rangle\} \qquad 174c$$

$$\xrightarrow{\text{index expansion}}$$

$$[owns,ref2^{\smile}donkey\times 0,ref1^{\smile}man\times 0]$$
$$[owns,ref2^{\smile}donkey\times 1,ref1^{\smile}man\times 1]$$
$$\{\langle\{ref1\times 0\},\{e3\}\rangle,\langle\{ref2\times 0\},\{e5\}\rangle,\langle\{ref1\times 1\},\{e4\}\rangle,\langle\{ref2\times 1\},\{e6\}\rangle\}$$
$$\{e1 \gg e3, e1 \gg e4, e2 \gg e5, e2 \gg e6\} \qquad 174d$$

$$\xrightarrow{\text{renaming}}$$

$$[owns,ref6^{\smile}donkey,ref4^{\smile}man]$$
$$[owns,ref5^{\smile}donkey,ref3^{\smile}man]$$
$$\{\langle\{ref4\},\{e3\}\rangle,\langle\{ref6\},\{e5\}\rangle,\langle\{ref3\},\{e4\}\rangle,\langle\{ref5\},\{e6\}\rangle\}$$
$$\{e1 \gg e3, e1 \gg e4, e2 \gg e5, e2 \gg e6\} \qquad 174e$$

6.6. More Complicated Readings: Non-Uniform Quantification

The are two more readings for which we must account. Both of them arise where the quantification in the sentence is in a sense non-uniform. By this, I mean that not all the members of the sets participating in the quantified relation take part in the same way; in this two man/two donkey example, say, if one donkey is owned by one man, and another is owned by two. In general, there are three ways this can happen, distinct from the readings generated in the earlier sections; only two arise in this simple example, so the third will be deferred for now.

The first of these readings, shown in figure 13d, arises when there is exactly the number of participants in the relation as is stated in the surface form, but the men own different numbers of donkeys (as in the figure). Note that this reading is unusual, in that it is the only reading where the predicate does not range over the entirety of both sets. For this reason, it is not derivable under index propagation, partition and/or expansion as defined so far, because they can only generate readings by producing members or subsets of sets, and do not allow some members of a set to be excluded. Therefore, this large class of readings will require a special mechanism. The second reading, which Figure 13e illustrates, can arise where each man owns a (different) subset of the complete set of donkeys, the cardinality of which is given in the surface form, but the subsets intersect. The required GRL representations of these readings are shown in 175 and 176 respectively.

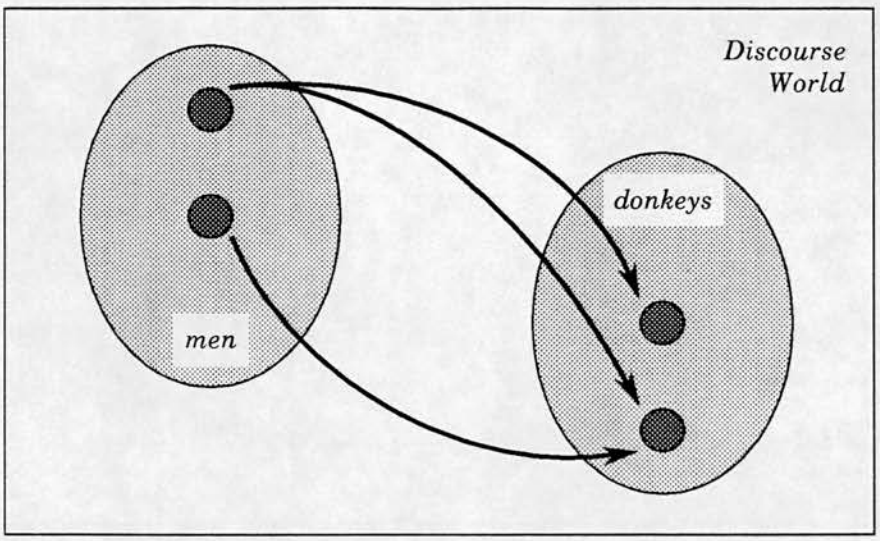


Figure 13d: *One man owns two donkeys; another owns one of them.*

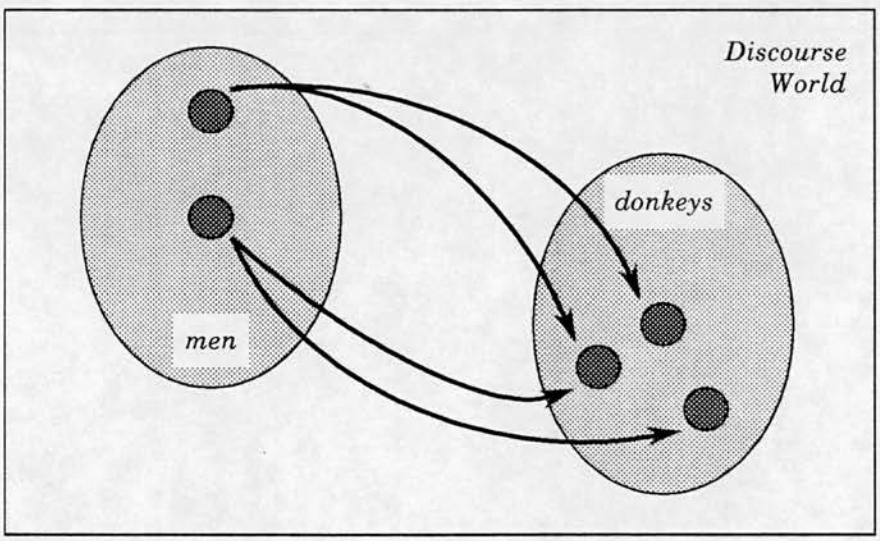


Figure 13e: *Each of two men owns one of an intersecting pair of sets of donkeys.*

[owns,ref5˜donkey,ref3˜man] [owns,ref5˜donkey,ref4˜man]
[owns,ref6˜donkey,ref4˜man]
$\{\langle\{ref5\},\{e5\}\rangle,\langle\{ref3\},\{e3\}\rangle,\langle\{ref6\},\{e6\}\rangle,\langle\{ref4\},\{e4\}\rangle\}$
$\{e1 \gg e3, e1 \gg e4, e2 \gg e5, e2 \gg e6\}$ 175

[owns,ref5˜donkey,ref3˜man] [owns,ref6˜donkey,ref3˜man]
[owns,ref6˜donkey,ref4˜man] [owns,ref7˜donkey,ref4˜man]
$\{\langle\{ref5\},\{e5\}\rangle,\langle\{ref3\},\{e3\}\rangle,\langle\{ref6\},\{e6\}\rangle,\langle\{ref7\},\{e7\}\rangle,\langle\{ref4\},\{e4\}\rangle\}$
$\{e1 \gg e3, e1 \gg e4, e2 \gg e5, e2 \gg e6, e2 \gg e7\}$ 176

In order to generate these more obscure readings, we will first need to define a further operation, *Weak Index Partition*, which will enable us to split the references and entity tokens into smaller sections, and produce the meaning of the whole in terms of the parts, in standard divide-and-conquer style. Weak Index Partition (in fact Index Partition in general) is the linguistic analogue of the Entity Token Partition operation defined before on George's intensional representation of the discourse world.

Inspection of the various translations presented in this section shows that they do indeed represent the readings of this sentence we need. However, it is all too obvious that, while they perform the function we need for this simple sentence, they cease to produce all the readings we need as soon as we extend the size of our sets. Therefore, we must generalise the functions. First, though, there is an important point to be made.

6.7. Choosing Index Operations

The issue is this. We have defined three index operations, expansion, propagation and dependency, which we can try to apply to each reference in an expression, but we have no decision procedure for choosing between them. We will apparently need such a decision procedure at some stage, since it does not make sense for more than one sequence of these index operations to apply to a reference at once – this would mean that we could give two distinct correct readings of the vagueness in underspecified sentences at the same time, which is never true. What is more (and this becomes worse in the more general examples below) we can run into a seriously explosive situation if we try to make all the possible expansions in parallel in different possible translations. The Index Partition operations specified here, in conjunction with entity token partition will give us the means to propose a representation based on that already used in George for sets, which will considerably alleviate the burden of notating such an explosion, by allowing us to reduce the adaptability of our GRL expressions, without doing so completely. Even so notation cannot in general help us with the problem of searching through a potentially combinatorially increasing search space.

However, the problem is in practice not as serious as it might first appear. While it is certainly true that the potential for this explosion is very likely to occur in real discourse, it is nevertheless rarely (if ever) realised. The tendency is for reasoning to proceed at an abstract level (the level of sets) until complete grounding of the abstraction (to enumerated individuals) is absolutely required – for example, picking out one member of a

set by a singular reference need not lead to enumeration of all the members of that set. The distinction between references and entity tokens allows us to capture this abstraction.

What is more (and this is very relevant to George's strictly incremental, adaptable approach to discourse analysis) is that it is very unusual in English to cause such an enumeration in one statement. Either we have to quantify (*eg* "All of them", "Some of them", *etc*), which need not cause enumeration in George, or pick out subsets or individuals; in the subset and individual cases, each new reference generally only adds one step, or maybe a few steps, in the inference process from set to enumerated individuals, in model divide-and-conquer style. Thus, to a considerable extent, the work is done for us by the speaker in his/her use of language. Beyond this, for our purposes here, it is enough to say that we need some oracle to choose the best path through a search space.

Having made this point, we can go on relatively safely to consider the generalisation of the basic ideas presented above to a wider framework.

6.8. Generalising Weak Indexing

While the example concerning two men and two donkeys was useful to give an idea of the concepts involved here, it was lacking in that the two-member sets have some special properties, which are not true of sets in general. We now need to broaden our coverage to an example whose sets do not have these properties. Consider the example sentence and translation (before and after propagation) in 177.

"Three men own three donkeys." 177a

$\forall ind1 < 3.\forall ind2 < 3.[owns,ref2^{\smile}donkey \times ind2,ref1^{\smile}man \times ind1]$ 177b

$\forall ind1 < 3.\forall ind2 < 3.[owns,ref2^{\smile}donkey \times ind2 \times ind1,ref1^{\smile}man \times ind1]$ 177c

Applying expansion directly to 177b gives us the reading where all the donkeys are owned by all the men, exactly as before. Applying index propagation and then index expansion yields the other extreme, where each of the three men owns his own set of three donkeys, again as before. Indeed, a little consideration of the mathematical nature of this procedure soon suggests that these two correct readings will be obtained for any combination of upper bounds. As in the previous example, index dependency (after index propagation and before expansion) yields the one-to-one correspondence result painlessly – though the informal definition given before is not fully general, as explained below. I have not yet defined a mechanism for producing the other more complicated possible readings

available in this situation (*eg* one man owns one donkey, the other two both own both of a further pair of donkeys).

In the analysis of example 170, the various possibilities for mapping of relations between individual references could be divided into three groups, according to the cardinalities of the sets involved. There was one reading where there were two men and four donkeys. There were three readings (one of which we have yet to generate) where there were two men and two donkeys. Finally, there was one reading (which, also, has yet to be covered), where there were two men, and some number of donkeys (in this case three) between the minimal (two) and the maximal (four) sets. These different readings can also be categorised in terms of the number and nature of divisions of sets they connote. The first, for example, involves simply splitting the set of men once to give a relation between each man and a set of donkeys; the last one involves splitting the men once, splitting the donkeys twice, and establishing relations between each man and one of two distinct single donkeys and then between both men and a third donkey. In the next section, this notion of splitting sets will give rise to the means of dealing with the two harder dependencies.

It is in the nature of the expansion and propagation operations that they extend trivially to larger sets. Let us now consider the status of index dependency, in the same circumstance. By definition, the one-to-one correspondence reading produced by index dependency is only possible when the upper bounds of the two quantifiers are equal, as they have been in the two examples so far. (We may sometimes be in a position to use this fact for inference when only one upper bound is known.) Therefore, we must be able to restrict the application of the operation to cases where this is true. To do so, I introduce a monadic operator, *upb*, on index symbols, which will return the maximum value attainable by the index given as its argument (or, in the event that such a value is not immediately available, a variable which can later be associated with it through unification). This operator corresponds with the mapping IndexRange in the definition of the Discourse State in Chapter 4. Armed with it, we can express index dependency in the full generality shown in Rule 12, and, for the first time, create a Bounding Constraint (an arithmetic equation between upper bounds of indices).

We can now in general generate the three more obvious readings of our underspecified sentences in a straightforward mechanical way. However, we have yet to answer three fundamental questions – first, what about the more complicated readings? – second, how do we deal with sets which are too large to enumerate practicably in this way – and, third, how do we deal with the central issue here, that of underspecified sets?

Let I_1 and I_2 be index symbols respectively, R be a simple reference, $R_1...R_n$, $S_1...S_m$ be references (n, m $\in \mathbb{N} \geq 0$), and let P be a predicate symbol. Then:

To express index dependency, we can rewrite an expression of the form

$$\forall I_1.\forall I_2.[P,R_n...R_1,R \times I_2 \times I_1,S_m...S_1]$$

by replacing $R \times I_2 \times I_1$ with $R \times I_1$ iff (upb I_1 = upb I_2) to give

$$\forall I_1.[P,R_n...R_1,R \times I_1,S_m...S_1]$$

This gives rise to the new Bounding Constraint

$$\text{upb } I_1 = \text{upb } I_2$$

Rule 12: *Index Dependency*

6.9. Weak Index Partition

Conveniently, using little more than the facilities already available to us, we can deal with all three problems using the same mechanism. In Chapter 6, I introduced the idea of *entity token partition*, to allow us to represent newly distinguished subsets of previously homogenous sets while still capturing the common origin of the subsets. The solution to the questions posed above lies in the extension of this idea to *weak index partition* – that is, the explicit partition of ranges of indices to restrict the expressions produced from them through the various index manipulations explained in this chapter; such a restriction corresponds with a reduction in the vagueness in a quantified expression.

As before, this idea follows the intuition that reasoning about sets at the level of intensions (or arbitrary objects, or prototypes) can often remove the need to consider their cardinality. Examples of this were given in Chapter 1, where sets were introduced, but never referred to as anything other than conglomerate individuals. Let us now return to the original example of this chapter, reproduced below, and consider how index partition works.

("Some men own some donkeys." 169a

$\forall ind1.\forall ind2.[owns,ref2^{\smile}donkey \times ind2,ref1^{\smile}man \times ind1]$ 169b)

To recapitulate our motivation here: remember that indefinite references like these are viewed as introductory. As such, we are always free initially to assert the unexpanded indexed translation, 169b, in our Discourse Memory – because, by definition, these sets

have never been divided. However, when we come to refer back to them, just as we may need to split up (*ie* partition) the entity tokens representing them in response to the arrival of new sort information, we may also need to partition the indices correspondingly to express information learned about the sets from subsequent references. For example, suppose we have the bindings and entity tokens for 169b, as shown in 178a and b.

$$\{\langle\{\forall ind1.ref1 \times ind1\},\{e1\}\rangle,\langle\{\forall ind2.ref2 \times ind2\},\{e2\}\rangle\} \quad 178a$$

$$\{e1\text{~}man, e2\text{~}donkey\} \quad 178b$$

and suppose we add sentence 179a (with the non-introductory reading of the number quantifier). The appearance of the indefinite quantifier is a clear indication that the reference is not to the entire set; a less stilted form might be "Five of the..." where it is even clearer that the reference is to a subset. Thus, part 3 of the top level dereferencing algorithm is called into play; an attempt is made to find an existing entity token, bound singleton to an indexed reference whose sort is consistent with that of the five donkeys. The only one such is e2. e2 is therefore divided into two new entities, e3 and e4, one of which (the choice is arbitrary, because the two are identical) is bound to the new reference. The other is left unbound for the moment.

"Five donkeys escape." 179a

$$\forall ind3 < 5.[escape, ref3\text{~}donkey! \times ind3] \quad 179b$$

$$\{\langle\{\forall ind1.ref1\text{~}man \times ind1\},\{e1\}\rangle,\langle\{\forall ind2.ref2\text{~}donkey \times ind2\},\{e2\}\rangle,$$
$$\langle\{\forall ind3 < 5.ref3\text{~}donkey \times ind3\},\{e3\}\rangle\}$$
$$\{e1\text{~}man, e2\text{~}donkey, e2 \gg e3\text{~}donkey, e2 \gg e4\text{~}donkey\} \quad 179c$$

After analysis of this sentence, we are left with two disjoint subsets of the initial set of donkeys, one set of five, the other's size unknown, though related in the obvious way to that of the original. This information is shown in 179c; first, the bindings show ref3 bound to a new entity e3; then two new entities, e3 and e4 are shown, partition from e2 being denoted by $\gg$. Now, though we have no sort information which will definitionally set the absconding donkeys apart from the others, as led to entity token partition before, we do have a different distinguishing property of number. Number is fundamentally different from sort because it is a property only of the set of donkeys, and not of individuals in the set ([Haddock,1989] classes uniqueness of referents of definites as a "meta-constraint" – that is to say a constraint on the candidate set, which is the extension of a constraint set). Even so, the division produced by the application of a number property to a subset of a set is clearly analogous with that produced by application of sort properties, in as far as it requires creation of disjoint subsets, and thence entity token partition – otherwise, we

would be specifying two different sets with the same entity token, which is meaningless in the George framework.

Given this analogy, it makes sense to represent the division of the set in a way analogous with entity token partition. In this reference context, though, we do not need the historical aspects of that other kind of partition; references in George are only used once. Therefore, no partition tree structure (partial ordering $\gg$) is needed. Instead, we can divide the domain of a quantified reference between the two (or more) sets produced by the division. Since the index of an indexed reference directly represents the selection of a particular reference from its domain of quantification, the most intuitive way to do this is directly to divide that range. We already have a representation for this, in the form of association of upper bounds with indices.

It is important to state that this act of index partitioning is not strictly necessary in processing example 179. Indeed, it only really becomes necessary in certain subsequent circumstances (see below) – and therefore, in line with George's general "lazy" approach, we wish to defer it until it is necessary. To be more specific, introducing index partition on the initial introductory sentence will allow us to represent the new information we have about the whole set of donkeys – namely that there is one group of five, and another, which may or may not be empty (that point is debatable, but unimportant here). It does not tell us any more about the existing references as they stand. What is does do, though, is give us a convenient means of inferring information about the size of the original donkey set.

Suppose that we introduce the sentence shown in 180 (see Figure 14 for translation and bindings).

"The three donkeys who do not escape stay in the paddock." 180

and that we are asked the question 181

"How many donkeys are there?" 181

We now have a fully number-specified quantified reference to each of the entity tokens, e3 and e4, partitioned from e2, above (one in 179, one in 180). In response to question 181, we need to find a way of calculating the cardinality of the set, which is by definition equal to (*upb* ind2) in 178a. One way to do this is by performing index partition on expression 169b, so as to reflect the entity token partition which has already happened. The result of this operation (GRL expression, Bindings and Enities) is shown in 182.

$$\forall ind1.\forall ind2.[owns,ref2\text{~}donkey \times ind2, ref1\text{~}man \times ind1]$$

$$\text{—weak index partition}\rightarrow$$

$$\forall ind1.\forall ind4.[owns,ref2\text{~}donkey \times ind4, ref1\text{~}man \times ind1] \wedge$$
$$\forall ind1.\forall ind5.[owns,ref2\text{~}donkey \times ind5, ref1\text{~}man \times ind1]$$
$$\{\langle\{\forall ind1.ref1\text{~}man \times ind1\}, \{e1\}\rangle,$$
$$\langle\{\forall ind4.ref2 \times ind4\}, \{e3\}\rangle, \langle\{\forall ind5.ref2 \times ind5\}, \{e4\}\rangle\}$$
$$\{e1\text{~}man, e2\text{~}donkey, e2 \gg e3\text{~}donkey, e2 \gg e4\text{~}donkey\} \qquad 182$$

In doing so, we produce two new indexed references, whose upper bounds are (unknowns) M and N, such that (M + N) equals (*upb* ind2), because they are respectively bound to the two entity tokens which specify <u>all</u> the members of the set specified by the token from which they were partitioned. This equation constitutes a *Bounding Constraint*, as was defined in Chapter 4; as such, it must be added to the Discourse State. Now, Rule 4 (defined in Chapter 6) states that two indexed references singleton bound to the same entity token must have the equal upper bounds (expressing the fact that a given set can have only one – albeit unknown – cardinality). We can therefore infer that M and N equal 5 and 3 respectively, and thence, via the bounding constraint, that (*upb* ind2) equals 8.

This example, then, is one basic motivation for generalising the idea of index expansion to index partition – the former is the special case of the latter where all the indices have been partitioned until there is exactly one candidate discourse entity for each reference. The general form of index partition for weak indices is defined in Rule 13.

In the above example, we saw the top level view of a simple application of index partition. Let us now look at the actual mechanism of index partition, and show why the above example is the very simplest application of the idea. To recapitulate, the discourse so far is shown in 183.

"Some men own some donkeys.
Five donkeys escape.
The three donkeys who do not escape stay in the paddock." 183

Figure 14 shows the translation of the first three sentences of the example discourse and the corresponding bindings. (Note that the translation of "in the paddock" is not proposed as philosophically correct, but merely as adequate for experimental purposes.)

What we have at this stage, then, are George's usual collections of entity tokens and bindings. The entity tokens are informally divided into two kinds: a simple one each for the set of men and the paddock, and a compound one, specifying the whole set of donkeys,

Let I, $I_1...I_n$, $J_1...J_p$, $K_1...K_q$ be index symbols, R, $R_1...R_r$, $S_1...S_m$ be references ($n, m, p, q, r \in \mathbb{N}$), and P be a predicate symbol. Then:

We can rewrite an expression of the form

$$\forall J_1...\forall J_n.\forall I.\forall K_1...\forall K_q.[P,R_r,...,R_1,R \times I,S_m,...,S_1]$$

by dividing the domain of I into n parts, allowing new indices $I_1...I_n$ to range over them, and replacing the expression by n copies with I_i substituted for I, to give

$$\forall J_1...\forall J_n.\forall I_1.\forall K_1...\forall K_q.[P,R_r,...,R_1,R \times I_1,S_m,...,S_1]$$
$$\forall J_1...\forall J_n.\forall I_2.\forall K_1...\forall K_q.[P,R_r,...,R_1,R \times I_2,S_m,...,S_1]$$
$$...$$
$$\forall J_1...\forall J_n.\forall I_n.\forall K_1...\forall K_q.[P,R_r,...,R_1,R \times I_n,S_m,...,S_1]$$

This creates a new Bounding Constraint

$$\text{upb I} = \text{upb } I_1 + \text{upb } I_2 + ... + \text{upb } I_n$$

Whenever such an index partition is performed, a corresponding entity token partition must also be executed (if one does not already exist), and new bindings must be created to replace that of R × I.

Rule 13: *Weak Index Partition*

```
** s:∀ind1.∀ind2.[own(pres,pres,perf,act,indic),
                   ref2~donkey×ind2,
                   ref1~man×ind1]
** s:∀ind3<5.[escape(pres,pres,perf,act,indic),
                ref3~donkey!×ind3]
** s:∀ind4<3.[not escape(pres,pres,perf,act,indic),
                   ref4~donkey!×ind4] ▸
              [stay(pres,pres,perf,act,indic),
                   ref5~paddock!,
                   ref4~donkey!×ind4]

ref1 is bound to e1~man
ref2 is bound to e2~donkey
ref3 is bound to e3~donkey (e3 ⊆ e2)
ref4 is bound to e4~donkey (e4 ⊆ e2)
ref5 is bound to e5~paddock
```

Figure 14: *George output for the escaping donkey discourse.*

partitioned into two others specifying the absconded and present donkeys respectively. We have a set of bindings linking the tokens with the appropriate references in the translation of the discourse. Finally, we have a bounding constraint, representing the cardinality relationship between the divided set and its subsets.

When we applied index partition to the first sentence, we obtained this expression (where M, N stand for members of $\mathbb{N}$ such that $M+N = upb$ ind2 (returning to my usual shorthand notation for predicate symbols):

$$\forall ind1.\forall ind5 < M.[owns,ref2\tilde{}donkey \times ind5,ref1\tilde{}man \times ind1]$$
$$\forall ind1.\forall ind6 < N.[owns,ref2\tilde{}donkey \times ind6,ref1\tilde{}man \times ind1] \quad 184$$

Specifically, ind5 and ind6 were generated by renaming ind2 to new unique index symbols. M and N have been introduced to allow us to refer succinctly to *upb* ind5 and *upb* ind6. Recall that quantifier scope in GRL is such that all the quantifiers apply to both conjuncts; however, only ind1 affects both, because ind5 and ind6 are only applied (by the index application operator) to one conjunct each. Now, the references in this expression are all new, because by definition refN×X is unique, because, when X is instantiated to a constant according to the definition of index expansion, the resulting reference is renamed uniquely. When we analyse the reference in the expression, we can directly use the information presented to us in the tree representing the partitioned entity token, because it is built in to the top level dereferencing algorithm that existing entity tokens will be used instead of creating new ones wherever possible (in line with the Principle of Parsimony). We can therefore arbitrarily choose an entity token (because they are indistinguishable) specifying one of the subsets for each reference (in general, making the correct choice would involve a little search; this is not a problem, because the search space is tightly constrained by the sorts of the references and entities, and the existence of the entity token partition) and create the new bindings shown in 185:

$$\{\langle\{\forall ind5 < M.ref2 \times ind5\},\{e3\}\rangle,\langle\{\forall ind6 < N.ref2 \times ind6\},\{e4\}\rangle\} \quad 185$$

Now, this means that the new references are bound to the same entity tokens as ref3 and ref4, respectively. Furthermore, all the bindings are singleton. Therefore, we can infer that M is 5 and N is 3 (Rule 4 (Number Consistency) in Chapter 6), so the binding is now fully instantiated. Finally, for the moment, we must choose whether to replace the original translation of our first sentence with the partitioned version, and to replace the corresponding binding with the new ones, or whether to add in the new cardinality information to the quantifier of the old version (which can easily be done by the existing

unification matching in George, if we use a representation where unbounded indices are denoted as bounded by some uninstantiated variable).

6.10. Compound Index Manipulation

So far, this section has introduced the idea of weak indexing, the associated operations of weak index expansion and partition, index dependency and the idea of bounding constraints (which applies to both weak and strong indices). Weak indexing has only so far been discussed in terms of two-place predicates, both of whose arguments are weakly quantified; other possible combinations of arguments will be covered later, along with the equivalent strongly indexed combinations.

First, however, I must suspend our current example, while I fill in the gap in the foregoing exposition of weak index manipulation. This will complete our discussion of the interactions between weak indices.

Earlier, I presented the idea of using indices and the operations defined on them to express more or less of the detail of partially specified relations between members of sets in a discourse. I gave detail of how three possible readings of a sentence containing two underspecified set references can be generated. However, I also pointed out that there are harder readings, like those represented, for the two men/two donkey ownership situation, in Figures 13d and e. The figures are reproduced below.

The reason that generation of these two classes of reading has been left until now is that we can express the more complicated manipulations necessary more or less in terms of the other simple ones, if we are allowed the use of index partition, which had not been defined before. I will call the new, more complex operation defined here *compound index partition* to express the fact that it is made up of a number of other operations. As well as covering the two readings illustrated in the figures, I will explain a simple third situation not explicitly mentioned before. This dependency arises when a relation exists between members of two sets in such a way that both existing sets can be divided so that no member of a resulting set stands in the relation to a member of more than one other resulting set. More on this later.

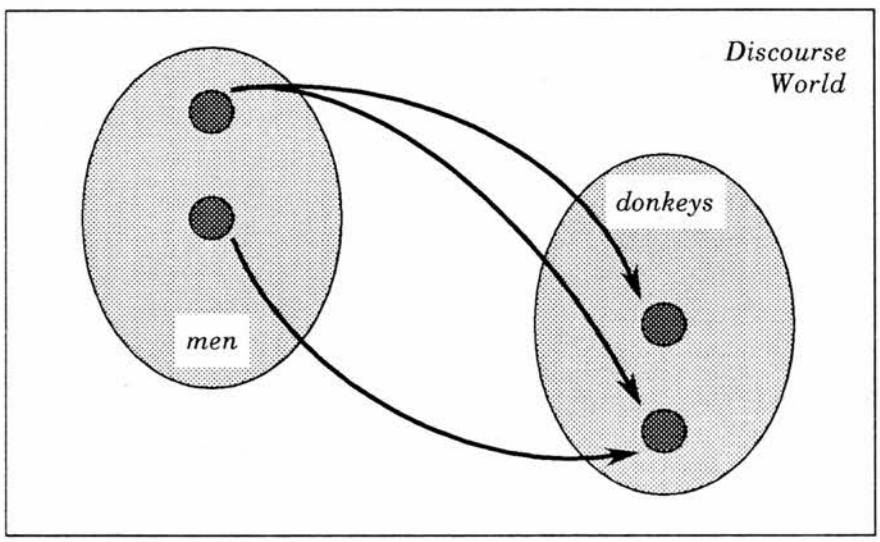
These manipulations may best be considered in the context of the input sentence given before in 177, which is reproduced for convenience below with its unpropagated and propagated translations, respectively.

("Three men own three donkeys." 177a

∀ind1<3.∀ind2<3.[owns,ref2˜donkey×ind2,ref1˜man×ind1] 177b

∀ind1<3.∀ind2<3.[owns,ref2˜donkey×ind2×ind1,ref1˜man×ind1] 177c)

The first reading to cover here arises from situations in the same class as that expressed for the two men/two donkey sentence in Figure 13d (reproduced below): the situation



(Figure 13d: One man owns two donkeys; another owns one of them.)

where not all of the men own an equal number of donkeys, but the number of donkeys is that specified in the surface form of the sentence. In the two/two situation there was only one way in which this could happen, because there were only two donkeys to be owned – each man owned either one or two of them. In the current example, however, there are three men, three donkeys, and (therefore) a lot more possible combinations. For example, two men might own three of the donkeys, and one two of them; or one man might own one, another two, and a third three. Note, though, that the defining feature of this class of cases is that at least one of the men owns all the donkeys – if this were not the case, this would be an example of the third compound relationship, which is explained below.

The problem with representing the kind of relationship seen in Figure 13d in our existing framework is that not all the donkeys are in the relation with respect to all the men. That is to say, some donkeys may be owned by one man, and some may be owned by all three. With the existing index manipulations we can only represent relationships where the relation applies to every member of both sets – *ie* all the donkeys were owned by all the men, or each donkey was owned by exactly one man, and so on. One point to note is that all

the donkeys must stand in the relation to at least one man, and *vice versa*; otherwise the sentence is no longer true of the sets in any reading.

The key to solving this issue is very similar to that in the earlier, simpler worked examples, with the difference that sometimes we will need to allow some entity tokens, produced in response to index partition, not to be included in the relation. This is a rather dangerous facility to have around – one can imagine, for example, arriving at readings where a predicate is only applied to half a set, and the other half is left unpredicated, which would be an incorrect interpretation of a sentence like, say, "The men run.". For this reason, we must consider the preconditions of this final weak index manipulation rule very carefully.

To be specific, the operation must only apply when two or more indexed references appear in the same sentence, as with our current example. Thus, it might be viewed as a kind of interaction between indices, like some of the other effects considered here; and therefore, it should be defined in terms of index propagation (Rule 11), or at least the index application rule (Rule 18), which govern such interaction. It should also be defined in terms of the existing index partition, rather than expansion – otherwise, it will not work on unbounded indices. Beyond this, the rule is fairly simple. Index partition can proceed roughly as before on both the sets, with its associated entity token partition. The difference in the new operation is that, for one or other of the two sets, some of the rewritten expressions may be discarded, so that some entity tokens are left without references to which to be bound. Thus, some elements of the set are left uncovered. This will become clear in the example below. The definition of Compound Index Partition is given in Rule 14.

For example, consider a reading of example 177 (reproduced below) where two out of three men both own all three donkeys, and third man owns just one. In GRL, we could write this as shown in 186a with the bindings given in 186b.

("Three men own three donkeys." 177a

$\forall ind1 < 3.\forall ind2 < 3.[owns,ref2^{\smile}donkey \times ind2,ref1^{\smile}man \times ind1]$ 177b

$\forall ind1 < 3.\forall ind2 < 3.[owns,ref2^{\smile}donkey \times ind2 \times ind1,ref1^{\smile}man \times ind1]$ 177c)

[owns,ref4˘donkey,ref1˘man]	[owns,ref4˘donkey,ref2˘man]
[owns,ref5˘donkey,ref1˘man]	[owns,ref5˘donkey,ref2˘man]
[owns,ref6˘donkey,ref1˘man]	[owns,ref6˘donkey,ref2˘man]

[owns,ref4˘donkey,ref3˘man] 186a

Let Q_1, Q_2 be (possibly empty) strings of quantifiers, I, $I_1...I_n$, J, $J_1...J_m$ be index symbols, R, $R_1...R_r$, S, $S_1...S_p$, $T_1...T_q$ be references (n, m, p, q, r $\in \mathbb{N}$), and P be a predicate symbol. Let k_i be in $\mathbb{N}$ for $1 \leq i \leq n$, such that $1 \leq k_i < m$. Then:

We can rewrite an expression of the form

$$Q_1.\forall I.\forall J.Q_2.[P,R_n,...,R_1,R \times I,S_p,...,S_1,S \times J,T_q...T_1]$$

iff the index application rule (Rule 18) holds of ⟨left,R × I,S × J⟩, as follows.

1. **Divide the domain of I into $I_1...I_n$ by weak index partition (Rule 18);**
2. **Divide the resulting expression containing I_1 by weak index partition of J into $J_1...J_m$;**
3. **Divide the domain of J in each remaining expression resulting from step 1, above, into the <u>same</u> index partition as in 2, replacing the expression by k_j copies with J_i substituted for J, to give (where $1 \leq i \leq n$ and $1 \leq k_j \leq m$)**

$$Q_1.\forall I_j.\forall J_1.Q_2.[P,R_n,...,R_1,R \times I_j,S_p,...,S_1,S \times J_1,T_q...T_1]$$
$$Q_1.\forall I_j.\forall J_2.Q_2.[P,R_n,...,R_1,R \times I_j,S_p,...,S_1,S \times J_2,T_q...T_1]$$
$$...$$
$$Q_1.\forall I_j.\forall J_{k_j}.Q_2.[P,R_n,...,R_1,R \times I_j,S_p,...,S_1,S \times J_{k_j},T_q...T_1]$$

New bindings must be created, to replace that of S × J, between $S \times J_2...S \times J_k$ and the 2nd...k_ith of the m new entity tokens, respectively.

The rule also applies in the reverse direction (ie with I and J interchanged).

Rule 14: *Compound Index Partition*

{⟨{ref1},{e1~man}⟩,⟨{ref2},{e2~man }⟩,⟨{ref3},{e3~man}⟩,
⟨{ref4},{e4~donkey}⟩,⟨{ref5},{e5~donkey}⟩,⟨{ref6},{e6~donkey}⟩} 186b

Suppose, now, that given the underspecified form, 177b, in an existing discourse, we subsequently analyse a sentence for which we need to derive a discourse state equivalent to 186 under the translation algorithm. One way this might happen is the introduction of a sentence starting "The man who owns one donkey..."; recall, again, that the processes of inference discussed here only take place when simpler forms of dereferencing have failed, according to the Principle of Parsimony. As I said before, it will often be the case (as in this example) that the reference causing the further elaboration of the underspecfied form will give some clues as to the shape that elaboration could take; which is just as well, since, in

general, the search space is combinatorial. To produce this particular expanded reading, we proceed as follows. I start with the original unpropagated sentence.

$$\forall ind1 < 3.\forall ind2 < 3.[owns,ref2^{-}donkey \times ind2,ref1^{-}man \times ind1]$$
$$\{\langle\{\forall ind1 < 3.ref1 \times ind1\},\{e1^{-}man\}\rangle,$$
$$\langle\{\forall ind2 < 3.ref2 \times ind2\},\{e2^{-}donkey\}\rangle\} \qquad 187a$$

First, we perform compound index partition on ind1 and ind2. This can be divided into two separate weak index partitions. The first runs like this. We partition ind1 (into ind3 and ind4), and perform the corresponding entity token partition on e1 (into e3 and e4) to give 187b. The translation includes a bounding contraint connecting the old index to the new ones. For clarity, I have substituted values for the upper bounds in the GRL expressions, as though by unification.

$$\forall ind3 < 2.\forall ind2 < 3.[owns,ref2^{-}donkey \times ind2,ref1^{-}man \times ind3]$$
$$\forall ind4 < 1.\forall ind2 < 3.[owns,ref2^{-}donkey \times ind2,ref1^{-}man \times ind4]$$
$$\{\langle\{\forall ind3 < 2.ref1 \times ind3\},\{e3^{-}man\}\rangle,\langle\{\forall ind4 < 1.ref1 \times ind4\},\{e4^{-}man\}\rangle,$$
$$\langle\{\forall ind2 < 3.ref2 \times ind2\},\{e2^{-}donkey\}\rangle\}$$
$$upb\ ind1 = upb\ ind3 + upb\ ind4 \qquad 187b$$

The second weak index partition applied as part of the compound operation is on ind2 in the first expression resulting from the first partition above, to give (including a new bounding constraint):

$$\forall ind3 < 2.\forall ind5 < 1.[owns,ref2^{-}donkey \times ind5,ref1^{-}man \times ind3]$$
$$\forall ind3 < 2.\forall ind6 < 2.[owns,ref2^{-}donkey \times ind6,ref1^{-}man \times ind3]$$
$$\forall ind4 < 1.\forall ind2 < 3.[owns,ref2^{-}donkey \times ind2,ref1^{-}man \times ind4]$$
$$\{\langle\{\forall ind3 < 2.ref1 \times ind3\},\{e3^{-}man\}\rangle,\langle\{\forall ind4 < 1.ref1 \times ind4\},\{e4^{-}man\}\rangle,$$
$$\langle\{\forall ind2 < 3.ref2 \times ind2\},\{e2^{-}donkey\}\rangle,$$
$$\langle\{\forall ind5 < 1.ref2 \times ind5\},\{e5^{-}donkey\}\rangle,\langle\{\forall ind6 < 2.ref2 \times ind6\},\{e6^{-}donkey\}\rangle\}$$
$$upb\ ind1 = upb\ ind3 + upb\ ind4$$
$$upb\ ind2 = upb\ ind5 + upb\ ind6 \qquad 187c$$

The final part of the operation requires us to apply the same index partition to ind2 in the third expression, but gives us the option of omitting any of the resulting expressions beyond the first. In this case, there is only one expression we can omit – the second – which leaves us with

$$\forall ind3 < 2.\forall ind5 < 1.[owns,ref2^{-}donkey \times ind5,ref1^{-}man \times ind3]$$
$$\forall ind3 < 2.\forall ind6 < 2.[owns,ref2^{-}donkey \times ind6,ref1^{-}man \times ind3]$$

$$\forall ind4<1.\forall ind5<1.[owns,ref2^{\smile}donkey\times ind5,ref1^{\smile}man\times ind4]$$
$$\{\langle\{\forall ind3<2.ref1\times ind3\},\{e3^{\smile}man\}\rangle,\langle\{\forall ind4<1.ref1\times ind4\},\{e4^{\smile}man\}\rangle,$$
$$\langle\{\forall ind5<1.ref2\times ind5\},\{e5^{\smile}donkey\}\rangle,\langle\{\forall ind6<2.ref2\times ind6\},\{e6^{\smile}donkey\}\rangle\}$$
$$upb\ ind1 = upb\ ind3 + upb\ ind4$$
$$upb\ ind2 = upb\ ind5 + upb\ ind6 \qquad 187d$$

The remaining derivation steps are all trivial index expansions. First ind4:

$$\forall ind3<2.\forall ind5<1.[owns,ref2^{\smile}donkey\times ind5,ref1^{\smile}man\times ind3]$$
$$\forall ind3<2.\forall ind6<2.[owns,ref2^{\smile}donkey\times ind6,ref1^{\smile}man\times ind3]$$
$$\forall ind5<1.[owns,ref2^{\smile}donkey\times ind5,ref3^{\smile}man]$$
$$\{\langle\{\forall ind3<2.ref1\times ind3\},\{e3^{\smile}man\}\rangle,\langle\{ref3\},\{e4^{\smile}man\}\rangle,$$
$$\langle\{\forall ind5<1.ref2\times ind5\},\{e5^{\smile}donkey\}\rangle,\langle\{\forall ind6<2.ref2\times ind6\},\{e6^{\smile}donkey\}\rangle\}$$
$$upb\ ind1 = upb\ ind3 + upb\ ind4$$
$$upb\ ind2 = upb\ ind5 + upb\ ind6 \qquad 187e$$

Next, expanding ind5 gives:

$$\forall ind3<2.[owns,ref4^{\smile}donkey,ref1^{\smile}man\times ind3]$$
$$\forall ind3<2.\forall ind6<2.[owns,ref2^{\smile}donkey\times ind6,ref1^{\smile}man\times ind3]$$
$$[owns,ref4^{\smile}donkey,ref3^{\smile}man]$$
$$\{\langle\{\forall ind3<2.ref1\times ind3\},\{e3^{\smile}man\}\rangle,\langle\{ref3\},\{e4^{\smile}man\}\rangle,$$
$$\langle\{ref4\},\{e5^{\smile}donkey\}\rangle,\langle\{\forall ind6<2.ref2\times ind6\},\{e6^{\smile}donkey\}\rangle\}$$
$$upb\ ind1 = upb\ ind3 + upb\ ind4$$
$$upb\ ind2 = upb\ ind5 + upb\ ind6 \qquad 187f$$

At this point, we have gone as far as we need to give suitable entity tokens (e4 and e5) to which the references in our new input ("The man who owns one donkey...") can be bound.

Later on in the discourse, however, we might well have to go further – maybe in response to a reference to "One of the men who owns three donkeys...". One way to do so would be to expand ind3, again renaming the resulting reference symbols and perform the corresponding partition on e3 (into e7 and e8) to give:

$$[owns,ref4^{\smile}donkey,ref5^{\smile}man] \qquad [owns,ref4^{\smile}donkey,ref6^{\smile}man]$$
$$\forall ind6<2.[owns,ref2^{\smile}donkey\times ind6,ref5^{\smile}man]$$
$$\forall ind6<2.[owns,ref2^{\smile}donkey\times ind6,ref6^{\smile}man]$$
$$[owns,ref4^{\smile}donkey,ref3^{\smile}man]$$
$$\{\langle\{ref5\},\{e7^{\smile}man\}\rangle,\langle\{ref6\},\{e8^{\smile}man\}\rangle,\langle\{ref3\},\{e4^{\smile}man\}\rangle,$$
$$\langle\{ref4\},\{e5^{\smile}donkey\}\rangle,\langle\{\forall ind6<2.ref2\times ind6\},\{e6^{\smile}donkey\}\rangle\}$$

$$upb\ \text{ind1} = upb\ \text{ind3} + upb\ \text{ind4}$$
$$upb\ \text{ind2} = upb\ \text{ind5} + upb\ \text{ind6} \qquad 187g$$

Finally (maybe in response to "Both the other men own each donkey..."), we expand ind2, renaming the resulting reference symbols and performing the corresponding partition on e6 (into e9 and e10) to give:

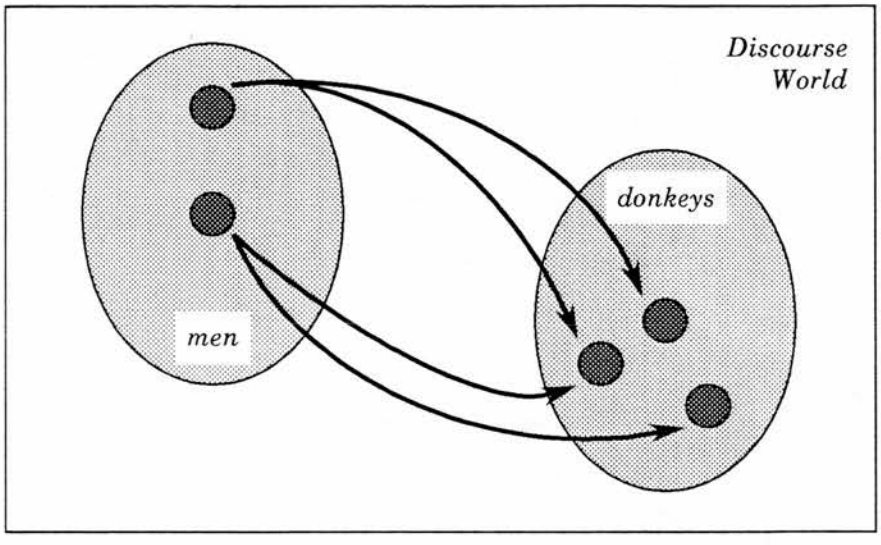
$$[\text{owns,ref4}^{\smile}\text{donkey,ref5}^{\smile}\text{man}] \qquad [\text{owns,ref4}^{\smile}\text{donkey,ref6}^{\smile}\text{man}]$$
$$[\text{owns,ref7}^{\smile}\text{donkey,ref5}^{\smile}\text{man}] \qquad [\text{owns,ref8}^{\smile}\text{donkey,ref5}^{\smile}\text{man}]$$
$$[\text{owns,ref7}^{\smile}\text{donkey,ref6}^{\smile}\text{man}] \qquad [\text{owns,ref8}^{\smile}\text{donkey,ref6}^{\smile}\text{man}]$$
$$[\text{owns,ref4}^{\smile}\text{donkey,ref3}^{\smile}\text{man}]$$
$$\{\langle\{\text{ref5}\},\{\text{e7}^{\smile}\text{man}\}\rangle,\langle\{\text{ref6}\},\{\text{e8}^{\smile}\text{man}\}\rangle,\langle\{\text{ref3}\},\{\text{e4}^{\smile}\text{man}\}\rangle,$$
$$\langle\{\text{ref4}\},\{\text{e5}^{\smile}\text{donkey}\}\rangle,\langle\{\text{ref7}\},\{\text{e9}^{\smile}\text{donkey}\}\rangle,\langle\{\text{ref8}\},\{\text{e10}^{\smile}\text{donkey}\}\rangle\}$$
$$upb\ \text{ind1} = upb\ \text{ind3} + upb\ \text{ind4}$$
$$upb\ \text{ind2} = upb\ \text{ind5} + upb\ \text{ind6} \qquad 187h$$

This is a renaming of expression 186, equivalent under the translation algorithm, which is what we were trying to derive. I emphasise again that these manipulations are intended to characterise the search space, and not (necessarily) to optimise search through it.

It is important to emphasise that this rule is not introduced *ad hoc*, just to produce the example reading above. Rather, it covers a very large class of readings (indeed, I will use it in the next class of example), where underspecification in set references is, in a sense, more extreme than in those readings where the correct specific translation can be produced by enumeration, as in the other rules. The rule is constrained to produce only those readings where a) all the individuals involved appear at least once in the relation, and b) where there is at least one individual whose part in the relation is not the same as that of all the others. Condition a) is enforced by part 1 of the rule, and condition b) is enforced by the upper bound on k_j in part 3. Thus, this rule cannot generate incorrect readings where some individuals do not appear at all in the relation (from a), and it cannot generate the same readings as the other rules (from b). Finally, the rule cannot apply to cases where there is not interaction between two or more set references, because it is defined in terms of the index application rule, which is by its own definition dependent on the presence of at least two indexed references.

Note also that this rule can introduce considerable non-determinism. Much of this non-determinism is encoded in the equational, arithmetic bounding constraints. Efficient, non-enumerating reasoning about such constraints is well within the capabilities of existing technology.

Let us now look at the situation represented (for two men/two donkeys) in Figure 13e (reproduced below). This situation is different from the ones covered before because the



(Figure 13e: Each of two men owns one of an intersecting pair of sets of donkeys.)

reference is such that there are actually more donkeys than are apparently specified in the sentence, but there is not a different set of donkeys for each man. The generality of this reading is a case where each man owns the same number of donkeys, but the sets of donkeys related to each man intersect in some arbitrarily complicated and unspecified way. For our example, let us take a case where there are actually five donkeys. One man owns the first three, the second man owns the second, third and fourth, and the third man owns the third, fourth, and fifth. This situation is explicitly represented in 188.

[owns,ref4~donkey,ref1~man] [owns,ref5~donkey,ref2~man]
[owns,ref5~donkey,ref1~man] [owns,ref6~donkey,ref2~man]
[owns,ref6~donkey,ref1~man] [owns,ref7~donkey,ref2~man]
[owns,ref6~donkey,ref3~man] [owns,ref7~donkey,ref3~man]
[owns,ref8~donkey,ref3~man] 188a

{⟨{ref1},{e1~man}⟩,⟨{ref2},{e2~man }⟩,⟨{ref3},{e3~man}⟩,
⟨{ref4},{e4~donkey}⟩,⟨{ref5},{e5~donkey}⟩,⟨{ref6},{e6~donkey}⟩,
⟨{ref7},{e7~donkey}⟩,⟨{ref8},{e8~donkey}⟩} 188b

The key to solving this issue lies in the description above: any such relation must be expressible as a collection of (different) relations between members of one set (the men) and a number of subsets of the other (the donkeys). In George, we already have a mechanism for converting our existing relation into such a collection of relations, by

dividing the sets between which the relations hold – namely, weak index partition, and the associated partition of entity tokens.

The motivation for the derivation which follows is as before: these inferences only take place when other attempts to dereference subsequent references have failed (according to the Principle of Parsimony, as encoded in the top level DeReferencing algorithm), and they only proceed until a situation has been reached where the current requirements of bindings and bounding constraints are satisfied. At each stage, I have underlined the parts of the expression to be affected by the next operation, to make the whole easier to read.

I start with the underspecified expression:

$$\forall ind1 < 3.\underline{\forall ind2 < 3.[owns,ref2^{\smile}donkey \times ind2,ref1^{\smile}man \times ind1]}$$
$$\{\langle\{\forall ind1 < 3.ref1 \times ind1\},\{e1^{\smile}man\}\rangle,$$
$$\underline{\langle\{\forall ind2 < 3.ref2 \times ind2\},\{e2^{\smile}donkey\}\rangle}\} \qquad 189a$$

The first step, here, is to partition ind2 (into ind3 and ind4). This expresses the fact that some donkeys stand in a different relationship to the men than others. Correspondingly, the entity token, e2, must be partitioned (into e3 and e4). The result is:

$$\forall ind1 < 3.\forall ind3 < 1.[owns,ref2^{\smile}donkey \times ind3,ref1^{\smile}man \times ind1]$$
$$\underline{\forall ind1 < 3.\forall ind4 < 2.[owns,ref2^{\smile}donkey \times ind4,ref1^{\smile}man \times ind1]}$$
$$\{\underline{\langle\{\forall ind1 < 3.ref1 \times ind1\},\{e1^{\smile}man\}\rangle},$$
$$\langle\{\forall ind3 < 1.ref2 \times ind3\},\{e3^{\smile}donkey\}\rangle,$$
$$\langle\{\forall ind4 < 2.ref2 \times ind4\},\{e4^{\smile}donkey\}\rangle\}$$
$$\{e2 \gg e3, e2 \gg e4\} \qquad 189b$$

Now a rather trivial expansion of ind3 (whose upper bound is unity) gives us a statement that one donkey (specified by e3) is owned by all three men (specified by e1) – maybe in response to the reference "The donkey which all the men own...".

$$\forall ind1 < 3.[owns,ref3^{\smile}donkey,ref1^{\smile}man \times ind1]$$
$$\underline{\forall ind1 < 3.\forall ind4 < 2.[owns,ref2^{\smile}donkey \times ind4,ref1^{\smile}man \times ind1]}$$
$$\{\underline{\langle\{\forall ind1 < 3.ref1 \times ind1\},\{e1^{\smile}man\}\rangle},\langle\{ref3\},\{e3^{\smile}donkey\}\rangle,$$
$$\langle\{\forall ind4 < 2.ref2 \times ind4\},\{e4^{\smile}donkey\}\rangle\}$$
$$\{e2 \gg e3, e2 \gg e4\} \qquad 189c$$

One possible next step is to compound partition ind1 with ind4 in the second expression. The first stage of the compound partition – partition of ind1 – leaves us with this intermediate result and new entity token partition.

$$\forall ind1 < 3.[owns,ref3^{\smile}donkey,ref1^{\smile}man \times ind1]$$
$$\underline{\forall ind5 < 2.\forall ind4 < 2.[owns,ref2^{\smile}donkey \times ind4,ref1^{\smile}man \times ind5]}$$
$$\underline{\forall ind6 < 1.\forall ind4 < 2.[owns,ref2^{\smile}donkey \times ind4,ref1^{\smile}man \times ind6]}$$
$$\{\langle\{\forall ind1 < 3.ref1 \times ind1\},\{e1^{\smile}man\}\rangle,\langle\{ref3\},\{e3^{\smile}donkey\}\rangle,$$
$$\langle\{\forall ind5 < 2.ref1 \times ind5\},\{e5^{\smile}man\}\rangle,\langle\{\forall ind6 < 1.ref1 \times ind6\},\{e6^{\smile}man\}\rangle,$$
$$\underline{\langle\{\forall ind4 < 2.ref2 \times ind4\},\{e4^{\smile}donkey\}\rangle}\}$$
$$\{e2 \gg e3, e2 \gg e4, e1 \gg e5, e1 \gg e6\} \qquad 189d$$

The second phase of the compound partition operation is to partition ind4 so that one of the resulting indices covers the whole of the ind1 partition above, while some of the others do not. One way of realising this is as follows.

$$\forall ind1 < 3.[owns,ref3^{\smile}donkey,ref1^{\smile}man \times ind1]$$
$$\forall ind5 < 2.\forall ind7 < 1.[owns,ref2^{\smile}donkey \times ind7,ref1^{\smile}man \times ind5]$$
$$\forall ind6 < 1.\forall ind7 < 1.[owns,ref2^{\smile}donkey \times ind7,ref1^{\smile}man \times ind6]$$
$$\underline{\forall ind5 < 2.\forall ind8 < 1.[owns,ref2^{\smile}donkey \times ind8,ref1^{\smile}man \times ind5]}$$
$$\{\langle\{\forall ind1 < 3.ref1 \times ind1\},\{e1^{\smile}man\}\rangle,\langle\{ref3\},\{e3^{\smile}donkey\}\rangle,$$
$$\langle\{\forall ind5 < 2.ref1 \times ind5\},\{e5^{\smile}man\}\rangle,\langle\{\forall ind6 < 1.ref1 \times ind6\},\{e6^{\smile}man\}\rangle,$$
$$\langle\{\forall ind7 < 1.ref2 \times ind7\},\{e7^{\smile}donkey\}\rangle,\underline{\langle\{\forall ind8 < 1.ref2 \times ind8\},\{e8^{\smile}donkey\}\rangle}\}$$
$$\{e2 \gg e3, e2 \gg e4, e1 \gg e5, e1 \gg e6, e4 \gg e7, e4 \gg e8\} \qquad 189e$$

We can now pick out another particular subset of donkeys: that whose members are owned by exactly one man. We do this first by expanding ind8 in the last expression.

$$\forall ind1 < 3.[owns,ref3^{\smile}donkey,ref1^{\smile}man \times ind1]$$
$$\forall ind5 < 2.\forall ind7 < 1.[owns,ref2^{\smile}donkey \times ind7,ref1^{\smile}man \times ind5]$$
$$\forall ind6 < 1.\forall ind7 < 1.[owns,ref2^{\smile}donkey \times ind7,ref1^{\smile}man \times ind6]$$
$$\underline{\forall ind5 < 2.[owns,ref4^{\smile}donkey,ref1^{\smile}man \times ind5]}$$
$$\{\langle\{\forall ind1 < 3.ref1 \times ind1\},\{e1^{\smile}man\}\rangle,\langle\{ref3\},\{e3^{\smile}donkey\}\rangle,$$
$$\langle\{\forall ind5 < 2.ref1 \times ind5\},\{e5^{\smile}man\}\rangle,\langle\{\forall ind6 < 1.ref1 \times ind6\},\{e6^{\smile}man\}\rangle,$$
$$\langle\{\forall ind7 < 1.ref2 \times ind7\},\{e7^{\smile}donkey\}\rangle,\underline{\langle\{ref4\},\{e8^{\smile}donkey\}\rangle}\}$$
$$\{e2 \gg e3, e2 \gg e4, e1 \gg e5, e1 \gg e6, e4 \gg e7, e4 \gg e8\} \qquad 189f$$

The next step is to propagate ind5 to ref4, giving 189g, expressing a one-to-one relation between a pair of men and a pair of donkeys not referred to elsewhere in this derivation.

$$\forall ind1 < 3.[owns, ref3^{\smile} donkey, ref1^{\smile} man \times ind1]$$
$$\underline{\forall ind5 < 2.\forall ind7 < 1.[owns, ref2^{\smile} donkey \times ind7, ref1^{\smile} man \times ind5]}$$
$$\forall ind6 < 1.\forall ind7 < 1.[owns, ref2^{\smile} donkey \times ind7, ref1^{\smile} man \times ind6]$$
$$\forall ind5 < 2.[owns, ref4^{\smile} donkey \times ind5, ref1^{\smile} man \times ind5]$$
$$\{\langle\{\forall ind1 < 3.ref1 \times ind1\}, \{e1^{\smile} man\}\rangle, \langle\{ref3\}, \{e3^{\smile} donkey\}\rangle,$$
$$\langle\{\forall ind5 < 2.ref1 \times ind5\}, \{e5^{\smile} man\}\rangle, \langle\{\forall ind6 < 1.ref1 \times ind6\}, \{e6^{\smile} man\}\rangle,$$
$$\underline{\langle\{\forall ind7 < 1.ref2 \times ind7\}, \{e7^{\smile} donkey\}\rangle}, \langle\{\forall ind5 < 2.ref4 \times ind5\}, \{e8^{\smile} donkey\}\rangle\}$$
$$\{e2 \gg e3, e2 \gg e4, e1 \gg e5, e1 \gg e6, e4 \gg e7, e4 \gg e8\} \qquad 189g$$

We can now expand all the indices upper bounded by unity and rename accordingly.

$$\forall ind1 < 3.[owns, ref3^{\smile} donkey, ref1^{\smile} man \times ind1]$$
$$\underline{\forall ind5 < 2.[owns, ref5^{\smile} donkey, ref1^{\smile} man \times ind5]}$$
$$\underline{[owns, ref5^{\smile} donkey, ref6^{\smile} man]}$$
$$\forall ind5 < 2.[owns, ref4^{\smile} donkey \times ind5, ref1^{\smile} man \times ind5]$$
$$\{\langle\{\forall ind1 < 3.ref1 \times ind1\}, \{e1^{\smile} man\}\rangle, \langle\{ref3\}, \{e3^{\smile} donkey\}\rangle,$$
$$\langle\{\forall ind5 < 2.ref1 \times ind5\}, \{e5^{\smile} man\}\rangle, \langle\{ref6\}, \{e6^{\smile} man\}\rangle,$$
$$\underline{\langle\{ref5\}, \{e7^{\smile} donkey\}\rangle}, \langle\{\forall ind5 < 2.ref4 \times ind5\}, \{e8^{\smile} donkey\}\rangle\}$$
$$\{e2 \gg e3, e2 \gg e4, e1 \gg e5, e1 \gg e6, e4 \gg e7, e4 \gg e8\} \qquad 189h$$

To complete the required relation, we wish to create a one-to-one ownership sub-relation between the set specified by e7 and that specified by e5, and an ownership sub-relation between the individual specified by e6 and the members of the set specified by e5. The mechanism existing in George to do this is index propagation (as used earlier in this derivation). Propagating ind5 to ref5 in the second expression gives the one-to-one relations; the resulting appearace of ind5 and a suitable quantifier in the third expression (to maintain consistency of ref5) gives the two-to-one relation.

$$\forall ind1 < 3.[owns, ref3^{\smile} donkey, ref1^{\smile} man \times ind1]$$
$$\underline{\forall ind5 < 2.[owns, ref5^{\smile} donkey \times ind5, ref1^{\smile} man \times ind5]}$$
$$\underline{\forall ind5 < 2.[owns, ref5^{\smile} donkey \times ind5, ref6^{\smile} man]}$$
$$\underline{\forall ind5 < 2.[owns, ref4^{\smile} donkey \times ind5, ref1^{\smile} man \times ind5]}$$
$$\{\langle\{\forall ind1 < 3.ref1 \times ind1\}, \{e1^{\smile} man\}\rangle, \langle\{ref3\}, \{e3^{\smile} donkey\}\rangle,$$
$$\langle\{\forall ind5 < 2.ref1 \times ind5\}, \{e5^{\smile} man\}\rangle, \langle\{ref6\}, \{e6^{\smile} man\}\rangle,$$
$$\underline{\langle\{\forall ind5 < 2.ref5 \times ind5\}, \{e7^{\smile} donkey\}\rangle}, \underline{\langle\{\forall ind5 < 2.ref4 \times ind5\}, \{e8^{\smile} donkey\}\rangle}\}$$
$$\{e2 \gg e3, e2 \gg e4, e1 \gg e5, e1 \gg e6, e4 \gg e7, e4 \gg e8\} \qquad 189i$$

The available operations are now much more tightly constrained. Next, we expand ind5 and perform corresponding partitions of e7 and e8.

$$\underline{\forall ind1 < 3.[owns,ref3^{\sim}donkey,ref1^{\sim}man \times ind1]}$$

$$[owns,ref9^{\sim}donkey,ref7^{\sim}man] \qquad [owns,ref10^{\sim}donkey,ref8^{\sim}man]$$

$$[owns,ref9^{\sim}donkey,ref6^{\sim}man] \qquad [owns,ref10^{\sim}donkey,ref6^{\sim}man]$$

$$[owns,ref11^{\sim}donkey,ref7^{\sim}man] \qquad [owns,ref12^{\sim}donkey,ref8^{\sim}man]$$

$$\{\langle\{\underline{\forall ind1 < 3.ref1 \times ind1}\},\{e1^{\sim}man\}\rangle,\langle\{ref3\},\{e3^{\sim}donkey\}\rangle,$$
$$\langle\{ref7\},\{e9^{\sim}man\}\rangle,\langle\{ref8\},\{e10^{\sim}man\}\rangle,\langle\{ref6\},\{e6^{\sim}man\}\rangle,$$
$$\langle\{ref9\},\{e11^{\sim}donkey\}\rangle,\langle\{ref10\},\{e12^{\sim}donkey\}\rangle,$$
$$\langle\{ref11\},\{e13^{\sim}donkey\}\rangle,\langle\{ref12\},\{e14^{\sim}donkey\}\rangle\}$$
$$\{e2 \gg e3, e2 \gg e4, e1 \gg e5, e1 \gg e6, e4 \gg e7, e4 \gg e8,$$
$$e5 \gg e9, e5 \gg e10, e7 \gg e11, e7 \gg e12, e8 \gg e13, e8 \gg e14\} \qquad 189j$$

Finally, we expand ind1, using the existing partition of e1, to give an expression with semantics identical with that of our target, 188, under the translation algorithm defined in Chapter 4.

$$[owns,ref3^{\sim}donkey,ref13^{\sim}man] \qquad [owns,ref3^{\sim}donkey,ref14^{\sim}man]$$
$$[owns,ref3^{\sim}donkey,ref15^{\sim}man]$$
$$[owns,ref9^{\sim}donkey,ref7^{\sim}man] \qquad [owns,ref10^{\sim}donkey,ref8^{\sim}man]$$
$$[owns,ref9^{\sim}donkey,ref6^{\sim}man] \qquad [owns,ref10^{\sim}donkey,ref6^{\sim}man]$$
$$[owns,ref11^{\sim}donkey,ref7^{\sim}man] \qquad [owns,ref12^{\sim}donkey,ref8^{\sim}man]$$

$$\{\langle\{ref13\},\{e6^{\sim}man\}\rangle,\langle\{ref14\},\{e9^{\sim}man\}\rangle,\langle\{ref15\},\{e10^{\sim}man\}\rangle,$$
$$\langle\{ref3\},\{e3^{\sim}donkey\}\rangle,\langle\{ref7\},\{e9^{\sim}man\}\rangle,\langle\{ref8\},\{e10^{\sim}man\}\rangle,$$
$$\langle\{ref6\},\{e6^{\sim}man\}\rangle,\langle\{ref9\},\{e11^{\sim}donkey\}\rangle,\langle\{ref10\},\{e12^{\sim}donkey\}\rangle,$$
$$\langle\{ref11\},\{e13^{\sim}donkey\}\rangle,\langle\{ref12\},\{e14^{\sim}donkey\}\rangle\}$$
$$\{e2 \gg e3, e2 \gg e4, e1 \gg e5, e1 \gg e6, e4 \gg e7, e4 \gg e8,$$
$$e5 \gg e9, e5 \gg e10, e7 \gg e11, e7 \gg e12, e8 \gg e13, e8 \gg e14\} \qquad 189k$$

Now, it is certainly the case that this derivation is extremely unwieldy. However, it must be borne in mind that the entirety of such a derivation would only happen in the most extreme cases in a discourse, where an underspecified reference is introduced, and then, subsequently, specified in full, bit by bit. It is unlikely, then, that such an extended sequence of operations would occur in a real discourse. Even if it did arise, sequences of inferences of this kind can only happen a few steps at a time, as more information becomes known about the relations in the sets. Therefore, given the adaptable nature of George's parsing, the system will never need to perform the search necessary to find the above

derivations in one go, because, as always, inference stops when a state has been found which is consistent with the reference of the discourse – *ie* in which no reference remains unbound.

The final kind of compound index manipulation is easier to imagine and to generate. It arises in the case where the sets are divisible into two separate relations. An example of this is given in Figure 16, where there is one man who owns exactly one donkey, and two

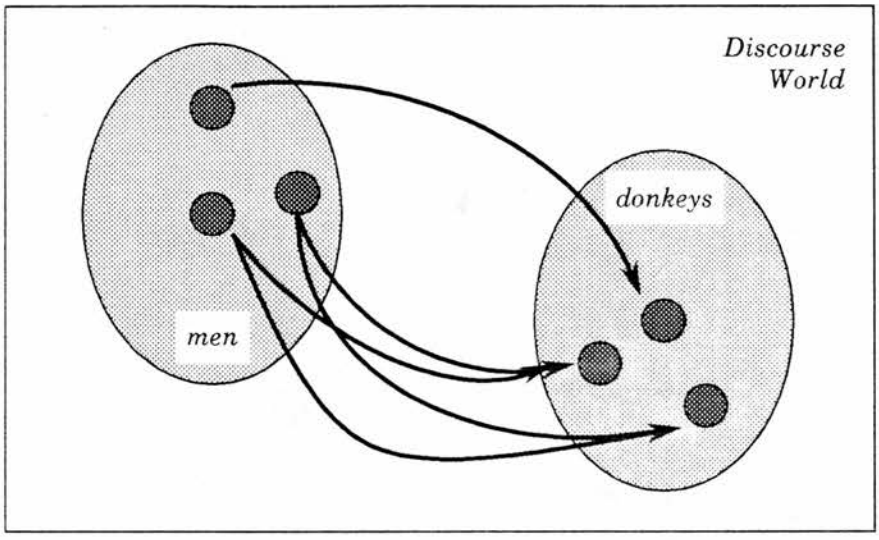


Figure 16: *"Three men own three donkeys."*

other men who own two other donkeys collectively. To represent this situation, we propagate and simultaneously partition the indices and entity tokens corresponding with the two sets, to give a conjunction of two new underspecified expressions. The references in the new expressions can then be dealt by recursive application of this and/or the other index dependencies.

7. Strong Indexing

In loose terms, strong indices, applied to the references which they index by the *strong index application operator*, ⊗, express a similar referential idea to that captured by weak indices – that is to say, the selection of individuals from the domain of quantification of a predicate. Strongly indexed references are used to express quantification where the quantifier (*eg* "each") gives a more constrained quantification than the weaker plurals, like "the". For example, 191 shows such a sentence and its translation. Let us suppose that the correct reading is that where every man owns exactly one donkey, but the particular donkey is unspecified, and no donkey is owned by more than one man. This is a valid

reading, so we need to generate it for some possible discourse; in this section, I will show how this is done. For clarity, this example includes the explicit unbounded index representation suggested above for "Each man", Q being the uninstantiated upper bound. The new sentence is presented in the context provided by the foregoing "escaping donkey" discourse, which is restated in 190. The George translation is given in Figure 14, which is reproduced below for convenience.

"Some men own some donkeys.
Five donkeys escape.
The three donkey who do not escape stay in the paddock." 190

```
** s:∀ind1.∀ind2.[own(pres,pres,perf,act,indic),
                    ref2~donkey×ind2,
                    ref1~man×ind1]
** s:∀ind3<5.[escape(pres,pres,perf,act,indic),
                  ref3~donkey!×ind3]
** s:∀ind4<3.[not escape(pres,pres,perf,act,indic),
                    ref4~donkey!×ind4] ▸
              [stay(pres,pres,perf,act,indic),
                      ref5~paddock!,
                      ref4~donkey!×ind4]

ref1 is bound to e1~man
ref2 is bound to e2~donkey
ref3 is bound to e3~donkey (e3 ⊆ e2)
ref4 is bound to e4~donkey (e4 ⊆ e2)
ref5 is bound to e5~paddock
```

(Figure 14: George output for the escaping donkey discourse.)

"Each man owns one donkey." 191a

$\forall ind7<Q.\forall ind8<1.[owns,ref7\tilde{}donkey\times ind8,ref6\tilde{}man!\otimes ind7]$ 191b

The point of this different representation is that it enables us to use a different set of operations over the indices. The full detail of this set is covered later (Rule 18ff), but for this example, we only need to know that strong indices always propagate leftwards on to indefinite references, as specified in Rule 15, unlike weak ones, which may or may not. After the propagation, we have a new logical expression as shown in 192:

$\forall ind7<Q.\forall ind8<1.[owns,ref7\tilde{}donkey\times ind8\otimes ind7,ref6\tilde{}man!\otimes ind7]$ 192

There is a further important difference between the behaviour of weak indices and strong ones. In the operations we have defined so far on weak indices, there is an assumption that

Let I be an index symbol, R, $R_1...R_n$, $S_1...Sm$ be references (n, m $\in \mathbb{N}$), and P be a predicate symbol. Then:

Given an expression of the form

$$\forall I.[P,R_n,...,R_1,R\otimes I,S_m,...,S_1],$$

all of R_i must be replaced with $R_i\otimes I$, and all of S_j with $S_j\otimes I$, iff

1) **the index I appears exactly once in the expression (ie it has not already been propagated); and**
2) **the Index Application Rule (Rule 18) holds for $\langle left,R_i,R\otimes I\rangle$ or $\langle right,S_j,R\otimes I\rangle$ as appropriate.**

To maintain consistency, I must also be propagated to any identical occurrences of R_i in other expressions in the discourse memory. If appropriate, quantifiers should be added.

Rule 15: *Strong Index Propagation*

manipulation of references corresponds one-to-one with manipulation of the entities in the discourse. This can see seen in the Rules 10 and 13, where entity token partition is introduced as part of the definition. For strong indices this is not the case. During the following examples, then, it should be borne in mind that when we expand strong indices we are creating multiple references and not multiple entity tokens; this notion will be formalised later. These references are free to be bound within the existing rules of the George system, subject to the correspondent entity token partition introduced in Rules 16 and 17.

Now, let us return to the running example, starting from sentence 169, and place the new sentence, 191 in that context. To analyse the reference of this sentence, there are four options potentially left open to us – namely weak index expansion and index partition of ind8, and some equivalent operations for strong indices on ind7. The option which leads to the derivation we need in this case is index expansion of ind8 (in 192), which gives us 193a as an intermediate stage. Subsequently, we rewrite ref7×0 with a new, unique reference symbol – here ref8 will do – to give 193b.

$$\forall ind7<M.[owns,ref7\text{`}donkey\times 0\otimes ind7,ref6\text{`}man!\otimes ind7] \quad 193a$$

$$\forall ind7<M.[owns,ref8\text{`}donkey\otimes ind7,ref6\text{`}man!\otimes ind7] \quad 193b$$

We now have a statement representing the fact that the relation between man references and donkey references is one to one. This behaviour would not have arisen from the weak indexed forms shown earlier, because of the numerous possible ways to generate relations between indexed references. The difference accords with the difference between strong and weak quantifiers in language (*eg* "each" and "the", respectively).

Now, in Chapter 6, Section 5.2, I mentioned that my assumption that all indefinites are introductory would cause problems later on. This is the first point at which it does so. For the purposes of explanation, therefore, I will from now on assume that non-introductory indefinite behaviour is available in George, as well as the existing introductory, giving explicit notice each time I do so. This is a reasonable step to take because the manipulations generating the various readings are not dependent on the introductory or non-introductory nature of the references and can be presented without reference to it.

When we analyse the reference of the expression, with our more general treatment of indefinites, we get the set of bindings shown in 194, where e1 and e2 are the entity tokens already existing in the discourse:

$$\{\langle\{\forall ind7 < M.ref6\check{}man!\otimes ind7\},\{e1\}\rangle,\langle\{\forall ind7 < M.ref8\check{}donkey\otimes ind7\},\{e2\}\rangle\} \quad 194$$

We know that the upper bound of one reference bound singleton to e2 is 8. Therefore, again applying Rule 4 (number consistency) to ref8⊗ind7, we can show that M is 8. Because M is the upper bound of both indices, the cardinality of e1 is now known to be 8. Therefore, for this reading, we can deduce the cardinality of the original set of men.

Even so, we have not yet needed to introduce a partition of that set of men. The final example in this sequence, 195, is a case where we need to introduce entity token partition in response to a weak index partition, rather than the other way round; in it, we need to use the partitioned expression directly to motivate the creation of new tokens. Note that the slight stiltedness of the language in this example is introduced to avoid the need for temporal reasoning, which is not covered in George. It does not affect the referential issues under discussion here.

"The men who own the donkeys who escape are angry." 195a

$$\forall ind9 < M.\forall ind10 < N.[escape,ref9\check{}donkey!\times ind10] \blacktriangleright$$
$$[own,ref9\check{}donkey!\times ind10,ref8\check{}man!\times ind9] \blacktriangleright$$
$$coref(\, ref8\check{}man!\times ind9,\, ref10\check{}angry!\,) \quad 195b$$

Now, we can deduce from this statement that there is a subset of the men who are angry, and that this subset is (from the previous sentence) in one-to-one correspondence with some subset of the set of donkeys – namely, the ones who escaped. George performs this deduction as follows, in the context of the preceding discourse shown in Figure 15.

```
** s:∀ind1<8.∀ind2<8.[own(pres,pres,perf,act,indic),
                        ref2~donkey×ind2,
                        ref1~man×ind1]
** s:∀ind3<5.[escape(pres,pres,perf,act,indic),
                     ref3~donkey!×ind3]
** s:∀ind4<3.[not escape(pres,pres,perf,act,indic),
                     ref4~donkey!×ind4] ▸
                  [stay(pres,pres,perf,act,indic),
                        ref5~paddock!,ref4~donkey!×ind4]
** s:∀ind7<8.[owns,ref8~donkey⊗ind7,ref6~man!⊗ind7]

ref1 is bound to e1~man
ref2 is bound to e2~donkey
ref3 is bound to e3~donkey (e3 ⊆ e2)
ref4 is bound to e4~donkey (e4 ⊆ e2)
ref5 is bound to e5~paddock
ref6 is bound to e1~man
ref8 is bound to e2~donkey
```

Figure 15: *George output for the escaping donkey discourse.*

The first few attempts, as George reads the initial words, to dereference "the men" in 195 will merely find the obvious binding with e1, because the context extension associated with the relative clause "who own the donkeys..." is not yet closed. Only when we reach "the donkeys" do we get a complete context extension about which we can reason. At this stage, the references to be analysed are as follows.

$$\{\forall ind9<M.\forall ind10<N.[own,ref9\tilde{}donkey!\times ind10,ref8\tilde{}man!\times ind9] \blacktriangleright ref8\tilde{}man!\times ind9,\ \forall ind10<N.ref9\tilde{}donkey!\times ind10\} \qquad 196$$

Following the dereferencing algorithm explained before, we arrive at these bindings:

$$\{\langle\{\forall ind9<8.\forall ind10<8.[own,ref9\tilde{}donkey!\times ind10,ref8\tilde{}man!\times ind9] \blacktriangleright ref8\tilde{}man!\times ind9\},\{e1\}\rangle,\ \langle\{\forall ind10<8.ref9\tilde{}donkey!\times ind10\},\{e2\}\rangle\} \qquad 197$$

Note that the upper bounds of the two indices have been instantiated during the context extension comparison, and that the precursor of the context extension here may be either

the first sentence or the last – the different quantification structures of the two sentences defining the ownership relation does not affect the context extension comparison algorithm (*qv*, Chapter 6).

The next point of interest in the parse is at the end of the second relative clause. We now have the following references to deal with:

$$\{\forall ind9 < M.\forall ind10 < N.[own,ref9\tilde{}donkey!\times ind10,ref8\tilde{}man!\times ind9] \blacktriangleright ref8\tilde{}man!\times ind9,$$
$$\forall ind10 < N.[escape,ref9\tilde{}donkey!\times ind10] \blacktriangleright ref9\tilde{}donkey!\times ind10\} \qquad 198$$

The reference to donkeys is now much more strongly constrained: in fact, it no longer refers to e2, but can only refer to e3. Therefore, the context extension matching algorithm leads us to infer that the original assumption that the "men" reference was to e1 was wrong, but that some subset of that specified by e1, in correspondence with that specified by e3, is a valid candidate. By the same reasoning, the new context extension rules out any existing entity token as a candidate for binding to ref6. The correct response to this is to attempt to find a statement about e3 which will allow the production of a suitable entity token. With the usual backwards search, the first expression found in the Discourse Memory is the strongly indexed ownership relation, 191. We know that e3 is a token partitioned from a token bound in this expression, from the partition tree in the Entity Tokens database. Therefore, it is possible that the expression can lead us to a partition of e1 such the references in our current utterance can be dereferenced. (Again, this higher level knowledge is external to George; the correct choice would really be found by simple search.) Therefore, we partition the indices in the sentence, guided in the obvious way by the existing partition of e2, so that the two "donkey" references in the resulting conjoined expression are bound to e3 and e4 respectively – that is to say, like this:

$$\forall ind11 < 5.[owns,ref8\tilde{}donkey!\otimes ind11,ref6\tilde{}man!\otimes ind11]$$
$$\forall ind12 < 3.[owns,ref8\tilde{}donkey!\otimes ind12,ref6\tilde{}man!\otimes ind12] \qquad 199$$

Now, we already have suitable entity tokens for ref8 to be bound to, but those for ref6 are missing. Therefore, as usual, we create some by entity token partition – which is a safe operation here, since it does not change the semantics of previous expressions in the system. Now, when we attempt to dereference "the men", we have a precursor in the first expression of 199. Therefore we can associate the set of men with the newly created entity token representing the five whose donkeys escaped. Finally, the coref operator in the main clause allows us to add the sort information "angry" to that entity token, thus completing the expression of the meaning of the discourse.

This extended work-through has shown a solution to the problem posed in Chapter 1, example 1. It has been useful because it shows the motivation behind this kind of analysis. However, in presenting it, I have left some gaps in the detail of the index manipulation operations, particularly with respect to strong indices. Therefore, I will now return to the more general level to complete the story.

8. Index Propagation on to Simple References

So far, I have only discussed the behaviour of index propagation on to quantified (indexed) references. For generality, though, and for full linguistic coverage, we need to discuss what happens when indices propagate to simple references.

Consider, first, sentence 200, in the context where there is already an entity token specifying a set of men, e1.

"The men own a donkey." 200a

$\forall ind1.[own,ref2^{\smile}donkey,ref1^{\smile}man!\times ind1]$ 200b

The bindings produced under the algorithms already explained are thus, where e2 is a newly introduced entity token specifying the donkey:

$$\{\langle\{\forall ind1.ref1^{\smile}man!\times ind1\},\{e1\}\rangle,\langle\{ref2^{\smile}donkey\},\{e2\}\rangle\} \quad 201$$

This translation represents the only correct reading of the example, where there is exactly one donkey and each man owns it. If we were to apply index propagation as before, we would produce a reading with a set of donkeys in one-to-one correspondence with the set of men. Therefore, weak index propagation onto simple indefinite references (such as this) is not allowed (see Rule 18).

If we now look at sentence 202 in the same way, we find that strong indexing produces a different effect.

"Each man owns a donkey." 202a

$\forall ind1.[own,ref2^{\smile}donkey,ref1^{\smile}man!\otimes ind1]$ 202b

The idea of weak index propagation was introduced earlier to allow us to express the existence of complicated relationships between quantified references and other kinds of reference. Here, we wish to express a corresponding intuition regarding the effect of strongly quantified references. However, the detail of the intuition is different. In this

strong index case, the effect of the quantifier on the indefinite reference is to produce, as it were, a set of references, one corresponding with each man, just as before. Unlike the weak index case, though, there is no associated requirement that each reference is bound to a different entity in the discourse – all the men could own the same donkey. Also, as in the treatment of example 191, we see the problem introduced by my simplifying assumption that indefinites are introductory, because, in many interesting readings, the indefinite noun phrase "a donkey" is non-introductory. As before, I will gloss over this shortcoming, and assume the non-introductory behaviour. Consider, for example, discourse 203:

"There are ten men and five donkeys. Each man owns a donkey." 203

The interesting point about this example is that the indefinite singular reference to "a donkey" in the second sentence of 203 refers to (a subset of) the entity introduced by the plural reference to "five donkeys" in the first sentence.

To capture this intuition in George we need a definition of index propagation for strong indices where the index always propagates to indefinite references, unlike weak indices. Using it, we can, as it were, make the surface form of the reference agree with the referent. We also need definitions of strong index expansion and partition, to enable us to rewrite the "shorthand" quantified expression as the more specific collection of expressions for which it stands. The rewriting must allow for different behaviour regarding entity token partition, from that associated with the weak index operations, because here that partition need not exactly correspond with the index partition – more than one man may own a given donkey.

Returning to example 202, then, after index propagation, we can produce the new translation and bindings shown in example 204, the appearance of the index on ref2 licensing its binding to e2 under Rule 4 (number consistency). I will define the strong index partition and expansion operations formally later in this chapter.

$$\forall ind1.[own,ref2^{\smile}donkey\otimes ind1,ref1^{\smile}man!\otimes ind1] \quad 204a$$

$$\{\langle\{\forall ind1.ref1^{\smile}man!\otimes ind1\},\{e1\}\rangle,\langle\{\forall ind1.ref2^{\smile}donkey\otimes ind1\},\{e2\}\rangle\} \quad 204b$$

Next, we must consider index propagation to simple definite references. Before, propagating of weak indices to a simple (*ie* singular) definite reference was unacceptable, because doing so would imply that the individual bound to the reference was in some sense divisible into elements in the same way as a set. This would suggest either that such a singleton-bound entity token was not really specifying a singleton in the first place, or

that it is possible arbitrarily to multiply singleton entities in a discourse. So neither possibility is true (or even sensible); therefore, we ruled out weak index propagation to simple definite references. The same reasoning shows that we do not want strong indices to propagate on to simple definite references. If they did so, we would be allowing a singular definite noun phrase to refer to several things at once, which is impossible.

There is one more set of cases to consider in this section. So far, I have considered index propagation only leftwards. The motivation for this is that weak quantification of the subject of a sentence may affect the quantification of the object(s), but not *vice versa* – recall the point made with the two men and two donkeys example at the beginning of this discussion: there can only ever be two men, even though there can be up to four donkeys. However, with strong quantifiers, this is not the case. Consider sentence 205; b is an intermediate unpropagated translation, and c is the final translation.

"A man owns every donkey." 205a

∀ind1.[owns,ref2⁻donkey!⊗ind1,ref1⁻man] 205b

∀ind1.[owns,ref2⁻donkey!⊗ind1,ref1⁻man⊗ind1] 205c

Here, as in the previous example, the intuition is that there is a different "man" reference (NB reference, not entity) for each donkey, so 205b fails to capture the meaning of the sentence. We need index propagation rightwards as well as leftwards, to give 205c, the correct translation. This accords with the notion that strong quantification is in some sense more virulent than the weak kind – in that strong quantification of a direct object can affect a subject, where weak quantification could not. Then, the expansion and partition rules for strong indices can be defined so that entity token partition takes care of associating the references with the correct entities in the discourse.

By the same reasoning as before, we will rule out rightwards propagations to definite references.

9. Sentences containing only Strong Indices

Just as I began this discussion with a sentence containing only weak indices, it will be useful to consider those containing only strong ones before we go on to look at mixed indices. Consider the sentence and translation shown in 206.

"Each man owns each donkey." 206a

∀ind1.∀ind2.[owns,ref2⁻donkey!⊗ind2,ref1⁻man!⊗ind1] 206b

Now, there is only one reading for this: that where the mapping is a cross product of the two sets. We can achieve this reading directly by an equivalent of the existing weak index expansion operation, with the same intuition: we view the indexed form as standing for some number of expressions which could state the relation in more detail; the expansion is then the operation of writing out this more detailed form. Note that the behaviour of these definite strongly indexed references is different from that in the indefinite cases in the last example. There, the number of entities in a set arising from a singular indefinite in the scope of a strong quantifier was indeterminate; here, there is no such vagueness.

Therefore, our expansion of the indices must be complete and symmetrical to give the correct reading, exactly as for the reference in 205 which introduced the quantifier. This leads me to suggest that strong indices do not interact with each other like weak ones. Therefore, strong index propagation on to strongly indexed references is not allowed in either direction, which accords with the intuition that strong quantification is somehow extreme – as though it had in some way used up all the "quantifiability" of the index which it applies.

Rule 16 is the definition of strong index expansion, where references are rewritten and renamed, as in weak index expansion, but the associated entity token partition may be only partial in the indefinite case (see Section 8). One noteworthy point is that index expansion of an indefinite strongly indexed reference has no associated entity token partition. The intermediate cases between complete entity token partition and no partition are covered in the strong index partition operation.

10. Sentences containing both Strong and Weak Indices

Sentences 207 and 208 characterise the only two ways in which a mixture of strong and weak indices can occur in two-place predications.

"Every man owns some donkeys." 207a

$\forall ind1.\forall ind2.[owns,ref2^{\smile}donkey \times ind2,ref1^{\smile}man! \otimes ind1]$ 207b

"Some men own every donkey." 208a

$\forall ind1.\forall ind2.[owns,ref2^{\smile}donkey! \otimes ind2,ref1^{\smile}man \times ind1]$ 208b

The exact readings we wish to derive here are where (in 207) each man in the set owns at least two donkeys, but it is known neither which donkeys he owns, nor if any or all of them

Let $I_1...I_p$, $J_1...J_q$ be index symbols, I be a bounded index symbol with upper bound N, and K be an arbitrary, possibly empty, string of I_i and J_i conjoined by $\times$. Let R, $R_1...R_n$, $S_1...S_m$ be references (n, m, p, q, $N \in \mathbb{N}$), and P be a predicate symbol. Then:

Given an expression of the form

$$\forall I_1...\forall I_p.\forall I<N.\forall J_1...\forall J_q.[P,R_n...R_1,R\times K\otimes I,S_m...S_1]$$

we can creating N copies of the expression, each with I replaced by a different member of $\mathbb{N}$ less than N, thus:

$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R\times K\otimes 0,S_m...S_1]$$
$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R\times K\otimes 1,S_m...S_1]$$
$$...$$
$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R\times K\otimes (N\text{-}1),S_m...S_1]$$

Then, if R is definite, we must partition its bound entity token so that there is a new, unique token bound to <u>each</u> copy of R; if R is indefinite, we must bind each new reference to the <u>single</u> existing entity token.

After this rewriting, each reference symbol with an integer applied to it must be replaced by a new reference symbol; all occurrences of each ⟨reference symbol,integer⟩ pair are replaced by the same symbol.

Rule 16: *Strong Index Expansion*

is owned by any other men and (in 208) each donkey is owned by at least two men, but it is not known which.

In these example, the index propagation rules give us the same propagation options as before. First, we propagate the strong index in 207 leftwards. This gives us:

$$\forall ind1.\forall ind2.[owns,ref2\check{}donkey\times ind2\otimes ind1,ref1\check{}man!\otimes ind1] \qquad 209$$

As before, on expansion or partition we require this to give us a set of donkeys for each man, the exact form of that set (*eg* disjoint, intersecting, equal) in relation to the others being unspecified.

In this example, leftward propagation of the weak index cannot apply because of the sentence structure – there is no argument leftwards to which it might propagate.

In example 208, one possible move would propagate a weak index left so that it dominated a strong one. As the terminology suggests, this is not acceptable – and, indeed, the readings it produces are erroneous, because they allow the generation of a different, disjoint set of donkeys for each man. Just allowing the alternative rightward propagation, where strong dominates weak, we can produce the reading shown in 210, again remembering that we must view the expansion of the indices as production of a set of references and not of a set of entities.

$$\forall ind1.\forall ind2.[owns,ref2^{-}donkey!\otimes ind2,ref1^{-}man\times ind1\otimes ind2] \qquad 210$$

Here, under strong index expansion, we have a set of men for each donkey, but no indication of which set is associated with which donkey, or if some sets of men own more than one of them. I will give a worked example of this process in Section 11.

By the same reasoning as before, when we replace the indefinite weakly quantified references in 207 and 208 with definite ones, we will no longer want strong indices to propagate. As before, this would imply that it were possible for a definite plural reference to refer to more than one set of things at once.

11. Strong Index Expansion and Partition

11.1. Introduction

Now that I have covered the possibilities of strong index propagation for every combination of kinds of reference, we have the data required to fill in the detail deferred before of the strong index equivalents of weak index expansion and partition.

The definitions of weak index partition and expansion include corresponding entity token partition. When we partition entity tokens to correspond with partition of weak indices, we are capturing the notion of dividing sets taking part in relationships, in order to describe the nature of those relationships more precisely, in a standard "divide and conquer" style. This idea extends to expansion from partition because expansion is nothing more than an extreme form of partition. Because of this division, we can justify the entity token partition – the act of weak index partition produces references which are non-coreferential, because it represents the division of sets into subsets.

When we have interaction between the two strengths of index, or between a strong index and a simple reference (which interaction could not arise with weak indices), we have a

rather different situation. The strong index is still certainly applying quantification to references, and thence to entity tokens, but the quantification seems to apply at a different level, or at least with a different emphasis.

11.2. Strong Index Expansion

Let us first look at strong index expansion, and consider the nature of the operation we are trying to define. The first thing to emphasise here, again, is the distinction between the divison of the sets specified by entity tokens into subsets or individuals, and the division of sets of references, as denoted by strong indexed references in George, into individual references.

In the definitions of weak index expansion and partition, the parallel partition of entity tokens represented the idea that the references being expanded and partitioned were manipulated at the same time and in the same way as the sets themselves. For the strong index operations, however, this is not always the case. In particular, expansion of strongly indexed indefinite references (generable, it seems, only by index propagation, and not by lexical lookup) does not always cause a parallel entity token partition, because (English) indefinite referrring expressions, within the scope of strong (English) quantifiers, do not exhibit this behaviour.

As a first approximation to a strong index expansion rule, then, we already know that strong quantifiers will not interact with other quantifiers in the way that weak ones will and that an associated entity token partition does not necessarily take place. Beyond this, we need to consider each possible kind of reference to which an index may be propagated, in order to find what action needs to be taken in the respective cases. However, it seems from the foregoing examples that the fundamental operation of strong index expansion is exactly the same as that of weak index expansion – that is, if the strong index has an upper bound, N (which may be supplied by reference to an existing bounded set), we can rewrite the quantified expression as a collection of N expressions, each with the quantifier removed and the index replaced by a distinct member of $\mathbb{N}$, less than N. The difference comes entirely in the behaviour of the entity tokens in response to this.

The only kinds of reference on to which strong index propagation may take place are the simple and weak indexed indefinite ones. However, strongly indexed definites can also arise directly from the input; these references always contain exactly one index.

First, let us deal with the simplest case, that of strongly indexed definite references. The intuition I am trying to capture is that the (English) strong quantification makes a relation true for each individual in its domain in isolation; no more subtle interaction between sets or collective interpretation (as in weak quantification) is allowed. The existing index expansion idea captures this in full; indeed, when we substitute members of $\mathbb{N}$ for indices and then rename reference symbols, we are doing more than just capturing the idea – we are performing exactly that operation. Therefore, when we expand a strongly indexed definite reference, we must decompose the entity token specifying the original set so each new reference has its own new entity token, just as for weak indices.

For indefinite strongly indexed references, the situation is different. When we expand such a reference, there is no associated entity token partition. This reflects the fact that, in 202, there may be as many donkeys as men, with each man owning a distinct one, or as few as one donkey, with each man owning it; the sentence is true, regardless.

("Each man owns a donkey." 202a

$\forall$ind1.[own,ref2˜donkey,ref1˜man!$\otimes$ind1] 202b)

Likewise, weakly indexed indefinites, once a strong index is applied, behave like definites. Again, we need to view the resultant reference as a set of references to some unspecified number of sets. However, this time the problem is more complicated. An example of this extra complication is found in sentence 211, where it is not only possible for the sets of donkeys to be non-distinct, but also for them to intersect.

"Every man owns some donkeys." 211a

$\forall$ind1.$\forall$ind2.[owns,ref2˜donkey$\times$ind2$\otimes$ind1,ref1˜man!$\otimes$ind1] 211b

This behaviour is represented in George by the existence of bindings to more than one entity token, where some entity tokens appear in more than one binding. Such a situation can be achieved in George by the application of (weak) index partition to the set of donkeys before the expansion. This leaves us with two or more sets of donkeys, each of which is in a different relation with the men (in particular, some are owned by more men than others). The kind of reasoning involved here, characterised as successive, alternate partitions of entity tokens and their bound references, was shown in full in the worked examples concerning weak index manipulations in Section 10. The formal definition of strong index expansion was given in Rule 16, in Section 9.

11.3. Strong Index Partition

The definition of strong index expansion is different from the weak index equivalent because of its alternative treatment of the entity token partition associated with the expansion operation. When we generalise expansion to partition, as we did with the equivalent weak index operations, we arrive at a rule which is similar to the expansion rule. The definition is shown in Rule 17.

Let I, $I_1...I_k$, $J_1...J_p$, $K_1...K_q$ be index symbols, R, $R_1...R_n$, $S_1...S_m$ be references ($n, m, p, q \in \mathbb{N}$), and P be a predicate symbol. Then:

We can rewrite an expression of the form

$$\forall J_1...\forall J_n.\forall I_1.\forall K_1...\forall K_q.[P,R_n,...,R_1,R\otimes I,S_m,...,S_1]$$

by dividing the domain of I into k parts, allowing new indices I_1 ... I_k to range over them, and replacing the expression by n copies with I_1 ... I_k respectively substituted for I, to give

$$\forall J_1...\forall J_n.\forall I_1.\forall K_1...\forall K_q.[P,R_n,...,R_1,R\otimes I_1,S_m,...,S_1]$$
$$\forall J_1...\forall J_n.\forall I_2.\forall K_1...\forall K_q.[P,R_n,...,R_1,R\otimes I_2,S_m,...,S_1]$$
$$...$$
$$\forall J_1...\forall J_n.\forall I_k.\forall K_1...\forall K_q.[P,R_n,...,R_1,R\otimes I_k,S_m,...,S_1]$$

This creates a new Bounding Constraint

$$\text{upb } I = \text{upb } I_1 + \text{upb } I_2 + ... + \text{upb } I_n$$

Whenever a strong index partition is performed upon a reference R, a corresponding entity token partition must also be executed on the token bound to it.

Rule 17: *Strong Index Partition*

12. Summary of Index Behaviour in Two-Place Predicates

The foregoing text, then, covers the behaviour of both strong and weak indices in all possible configurations within two-place closed predicates. In the following sections I will go on to cover their behaviour first in predicates of higher arity, and then in more complex

linguistic constructions, in particular where closed predicates are related by the context extension operator.

First, though, we can summarise the operations now available over the domain so far discussed with Table 1. The references between which propagation is carried out are identified in the table by the operators which define their relevant properties (in the case of simple indefinites, no identifier); a "–" in the propagation column means the operation is not applicable for pragmatic reasons (*ie* there is no index to propagate). It is useful to do this now, because we will see later that the information is also correct for the more complex linguistic input.

The table shows a pattern which we can encapsulate in three constraints, which define Rule 18, in conjunction with the index propagation operations defined in Rules 11 and 15.

Let R_i be references and I_j be indices. Let P be a predicate symbol, and D be a direction, such that $D \in \{ \text{left, right} \}$. Then:

An index may be propagated in a direction D from R_1 to R_2 (denoted by $\langle D, R_1, R_2 \rangle$) unless precluded by one of the following:

1. **No index may propagate on to a strongly indexed reference.**
2. **No weak index may propagate on to a simple reference.**
3. **No strong index may propagate on to a definite reference.**

Rule 18: *Index Application*

13. Index Behaviour in Closed Predicates with more than Two Arguments

13.1. Representing N-Place Predicates in GRL

In this section I will discuss the operation of index propagation in closed predicates containing more than two arguments. This will lead to a generalisation of some of the behaviour explained in the foregoing sections.

The function of predicates with more than two arguments in GRL is to allow representation of what we might call slot-fillers of verbs other than subject and object – for example, the indirect object in a verb like "to give". However, I will place a restriction on

Object (left) reference	Subject (right) reference	Rightward propagation	Leftward propagation
		–	–
!		–	–
×		illegal	–
⊗		compulsory	–
	!	–	–
!	!	–	–
×	!	illegal	–
⊗	!	illegal	–
	×	–	illegal
!	×	–	illegal
×	×	illegal	optional
⊗	×	compulsory	illegal
⊗	!×	illegal	illegal
	⊗	–	compulsory
!	⊗	–	illegal
×	⊗	illegal	compulsory
!×	⊗	illegal	illegal
⊗	⊗	illegal	illegal

Table 1: Applicability of Index Propagation in Two-Place Predicates

the kind of verbs covered in George – namely that only mono- and di-transitives can be represented using closed predicates of the appropriate arity – and let other instances of verb modification be covered by the context extension operator, as will be explained below. Indeed, there is an intuitive divide between the kinds of verb modifiers represented in George as extra arguments to closed predicates, and those represented by closed predicates with context extensions. This distinction will be covered in Chapter 9.

For the purposes of this section, then, we need to cover three-place predicates and context extended predicates. One-place predicates are not interesting in terms of propagation, since, by definition, it cannot take place within them.

In the next sections, I will show the utility of the intuitive division introduced above by highlighting different behaviour of these different kinds of information under index propagation.

13.2. Index Propagation to Indirect Objects

The first thing to note when we come to consider the propagation of indices to indirect objects is that the majority of di-transitive verbs are severely constrained in the readings they make available by the semantic nature of the events they describe. For example, the notion of a physical transfer of a single objects to each member of a set as in 190, connotes an ordering in time, which GRL cannot represent. Therefore, as before, I will constrain this discussion to producing those readings which are valid candidates as abstract translations, but which may subsequently be ruled out by real world inference.

"Every man gives the donkey the carrot." 190

We now need to consider the same combinations of propagations as before, but propagating weak and strong indices from subject and object on to the indirect object, and vice versa. The general detail of this, elaborated by the same reasoning as for two-place predicates, is identical with the two-place predicate version. For this reason, I will only explain here the differences, and suggest solutions to them.

As it happens (and as we would wish), the index propagation behaviour in two-place predicates is entirely subsumed by that in three-place; however, those properties of the general operation pertaining to propagation between non-adjacent arguments do not apply to two-place predicates (simply because there are no non-adjacent arguments in two place predicates). First, then, we must define the generality of propagation, and then the behaviour of the particular operators, both dominating and dominated, within that context. In particular, the behaviour of the strong index operator will need some refinement.

13.3. Sentences containing less than three indices

13.3.1. Introduction

First, consider some index propagating through a three-place predicate, of the form shown in 191 (where ind is applied by an indexing operator to one of { arg1 arg2 arg3 }).

∀ind.[pred,arg1,arg2,arg3] 191

Suppose, then, that the index appears on arg3 in the initial translation of the sentence. Then, we can move it to arg2, as before. Alternatively, we can move it to arg1; or to both of arg2 and arg1, remembering that each of these will be subject to constraints defined in terms of the kind of reference making up the argument. Since the respective behaviours of strong and weak indices are different, we will discuss them separately.

13.3.2. Strong Index Propagation in Three-Place Predicates

Consider sentence 192a, which, in the first instance, before any propagation has been applied, gives the translation shown in 192b:

"Each man offers a donkey a carrot" 192a

∀ind1.[offers,ref3ˇcarrot,ref2ˇdonkey,ref1ˇman!⊗ind1] 192b

∀ind1.[offers,ref3ˇcarrot⊗ind1,ref2ˇdonkey⊗ind1,ref1ˇman!⊗ind1] 192c

Now, in our definition of strong index propagation on two place predicates (Rule 15), we said that strong indices always propagate, subject to the index application rule, Rule 18. If we apply that propagation, we get expression 192c, which, as before, we can view as a shorthand form for some number of similar expressions. Strong index partition (Rule 17) can then give us the various readings available from this sentence, since, at each partition, any entity token can be viewed as specifying either one or a (separable) number of donkeys.

The readings correspond with sets of entities in the discourse world, like this:

N men, M donkey, P carrots, $1 \leq M \leq N$, $1 \leq P \leq N$.

Alternatively, the readings can be notated in FOPC using classical universal and existential quantifiers in the conventional way, thus:

$\exists c.carrot(c) \wedge \exists d.donkey(d) \wedge \forall m.man(m) \Rightarrow offers(c, d, m)$

$\exists c.carrot(c) \wedge \forall m.man(m) \Rightarrow \exists d.donkey(d) \wedge offers(c, d, m)$

$\exists d.donkey(d) \wedge \forall m.man(m) \Rightarrow \exists c.carrot(c) \wedge offers(c, d, m)$

$\forall m.man(m) \Rightarrow \exists c.carrot(c) \wedge \exists d.donkey(d) \wedge offers(c, d, m)$

This is to be expected. Strong index expansion is exactly like taking all possible assignments in a model of a quantified expression in FOPC; and this is the behaviour

required by strong quantification – a separate application of the predicate to each individual in the domain of quantification. Certainly, therefore, the theory embodied in George would be inadequate if it did not yield these readings. However, while the FOPC expressions above are adequate for expressing the behaviour of the references within an isolated sentence, they introduce problems if we attempt to extend the discourse beyond this one statement; also, manipulation of such expressions is less convenient than for their equivalents in GRL; I will discuss these problems fully in Chapter 8.

Returning to our discussion of strong index propagation in George: in the case where arg2 (in 191) is indexed, the same rules apply. Consider sentence 193a, translated (before compulsory index propagation) as 193b.

"A man offers every donkey a carrot." 193a

∀ind1.[offers,ref3⁻carrot,ref2⁻donkey!⊗ind1,ref1⁻man] 193b

The only reading available here (produced by leftward and rightward propagation) is represented in 194. Again, the sets of references are in one-to-one correspondence, by nature of the strong quantifier, though some or all of the references may be bound to the same entity token(s). Informally, this expresses the required idea that each donkey is associated with some carrot and some man by the relation, but exactly which carrot and which man is not specified.

∀ind1.[offers,ref3⁻carrot⊗ind1,ref2⁻donkey!⊗ind1,ref1⁻man⊗ind1] 194

Finally, in the case where arg1 (in 191) is indexed, we must propagate its index rightwards both one and two places, depending again on the legality of such an operation applied to the particular index operator(s) concerned. Consider 195.

"A man offers a donkey every carrot." 195a

∀ind1.[offers,ref3⁻carrot!⊗ind1,ref2⁻donkey,ref1⁻man] 195b

The finished translation here is shown in 196. Again, it denotes all the correct readings for the sentence, where some donkey and some man is associated with each carrot, but exactly which is not specified.

∀ind1.[offers,ref3⁻carrot!⊗ind1,ref2⁻donkey⊗ind1,ref1⁻man⊗ind1] 196

The complete rule for strong index propagation was given in Rule 15. It will become clear in the next section that extension of these ideas to sentences containing more than two quantifiers does not affect the behaviour of strong indices.

13.3.3. Weak Index Propagation in Three-Place Predicates

Let us now consider weak index propagation in three-place predicates by taking again sentence 191 and supposing the existence of a weakly indexed reference in each of the argument places. An example corresponding with this structure is given in 197 – all the references are this time weakly indexed, so that we may, according to the Index Application Rule (Rule 18), propagate any weak index leftwards on to them. By applying our existing rule for weak index propagation (Rule 11) to each argument position we can produce the readings shown in 197b-j – note that readings i and j result from different ordering of the same propagation operations; under the semantics given in Chapter 4, they are equivalent.

"Some men offer some donkeys some carrots" 197a

∀ind1.∀ind2.∀ind3.[offers,ref3˜carrot×ind3,
ref2˜donkey×ind2,ref1˜man×ind1] 197b

∀ind1.∀ind2.∀ind3.[offers,ref3˜carrot×ind3,
ref2˜donkey×ind2×ind1,ref1˜man×ind1] 197c

∀ind1.∀ind2.∀ind3.[offers,ref3˜carrot×ind3×ind1,
ref2˜donkey×ind2,ref1˜man×ind1] 197d

∀ind1.∀ind2.∀ind3.[offers,ref3˜carrot×ind3×ind1,
ref2˜donkey×ind2×ind1,ref1˜man×ind1] 197e

∀ind1.∀ind2.∀ind3.[offers,ref3˜carrot×ind3×ind2,
ref2˜donkey×ind2,ref1˜man×ind1] 197f

∀ind1.∀ind2.∀ind3.[offers,ref3˜carrot×ind3×ind2,
ref2˜donkey×ind2×ind1,ref1˜man×ind1] 197g

∀ind1.∀ind2.∀ind3.[offers,ref3˜carrot×ind3×ind2×ind1,
ref2˜donkey×ind2,ref1˜man×ind1] 197h

∀ind1.∀ind2.∀ind3.[offers,ref3˜carrot×ind3×ind2×ind1,
ref2˜donkey×ind2×ind1,ref1˜man×ind1] 197i

∀ind1.∀ind2.∀ind3.[offers,ref3˜carrot×ind3×ind1×ind2,
ref2˜donkey×ind2×ind1,ref1˜man×ind1] 197j

English paraphrases of these are as follows:

b: All the men offer all the donkeys all the carrots.

c: Certain men offer all the carrots to certain donkeys.

d: Certain men offer all the donkeys certain carrots.

e: Certain men offer certain donkeys carrots particular to the men.

f: All the men offer certain donkeys certain carrots.

g: Certain men offer certain donkeys carrots particular to the donkeys.

h: All the men offer all the donkeys carrots particular to both men and donkeys.

i,j: All the men offer certain donkeys carrots particular to both men and donkeys.

Note that the two readings i and j above, given by applications of the same set of index propagation operations in different orders, denote the same set of possible readings. This is because of the symmetrical nature of the index dependencies and the index expansion operation. In Chapter 4, such variants were defined to be equivalent.

An infinite number of different readings is available from these expressions, because the cardinality of the sets is not fixed; therefore, I will not enumerate them here. The important point is that the propagation operation is, as before, expressing a general dependency between the elements of the sets. Examination of the sentences above shows that they do indeed cover all the possible relations, on this general level: either the relationship denoted by the expression is simply between all the elements of a given two sets (where there is no propagation between them) or it is somehow more complicated – and must be dealt with by propagation, partition and/or index dependency.

As before, weak indices cannot propagate rightwards. This expresses the intuition that while sentence 192a can yield a unique set of donkeys and/or carrots for each man in the (single) set of men, the converse is not true: that is to say, the expression cannot give rise, for example, to a reading where there is one set of donkeys, each member of which is offered a carrot by a set of men unique to that donkey.

To summarise, weak quantification in three-place predicates behaves exactly as before in two-place predicates, even when there are more than two quantifiers in a sentence. Application of index dependency to references containing more than two indices will be covered in the next section.

13.4. Sentences containing more than two Indices

We are now able, subject to Rule 18 (index application) to produce GRL expressions containing references bearing three indices. Such expressions are always derived from input containing more than two quantified references within one clause.

In fact, the index application rule constrains the possibilities considerably – the only legal constructions are of three weak indices, or of one strong index dominating two weak ones.

We have already covered the behaviour of a strong index dominating a weak one, and of two weak indices together. Let us consider the possibilities, as before, to cover the full range of input data.

First, I will consider threefold weak indices, and index dependency. Consider the expression shown in example 198a. Supposing that the preconditions on the dependency of upper bound equality are not contradicted, we can rewrite this in two stages to give first 198b and then 198c. The rewrite could be performed in the opposite order, with no change in the resulting expression.

$$\forall ind1.\forall ind2.\forall ind3.[...,ref1 \times ind3 \times ind2 \times ind1,...] \quad 198a$$

$$\forall ind1.\forall ind2.[...,ref1 \times ind2 \times ind1,...] \quad 198b$$

$$\forall ind1.[...,ref1 \times ind1,...] \quad 198c$$

This new expression expresses a one-to-one correspondence between members of the three sets, exactly as in the twofold indexed version.

Finally for weak indices, note that the definition of multiply indexed references does not include an order on the indices. Thus, a single application of index dependency to 199a can give 199b, c or d, again assuming the precondition of equality of upper bound is fulfilled.

$$\forall ind1.\forall ind2.\forall ind3.[...,ref1 \times ind3 \times ind2 \times ind1,...] \quad 199a$$

$$\forall ind1.\forall ind2.[...,ref1 \times ind2 \times ind1,...] \quad 199b$$

$$\forall ind1.\forall ind3.[...,ref1 \times ind3 \times ind1,...] \quad 199c$$

$$\forall ind2.\forall ind3.[...,ref1 \times ind3 \times ind2,...] \quad 199d$$

These options express readings where two out of three sets are in one-to-one correspondence, and the third indexed reference denotes a set of individuals for each of the correspondent pairs in the other two sets.

When a strong index applies to two weak ones, we have a similar, though slightly simpler, situation. Index dependencies can apply to the two weak indices, as we would expect, but none can apply to the strong index – as always, this is expanded or partitioned directly, because that is the nature of strong quantifiers. Therefore, from an expression like 200b, the unpropagated translation of 200a, we can derive the reading in 200c. Together, b and c represent all the correct readings of the original sentence.

"Every man offers some donkeys some carrots." 200a

$$\forall ind1.\forall ind2.\forall ind3.[offers,ref3\text{`}carrot \times ind3 \times ind2 \otimes ind1,$$
$$ref2\text{`}donkey \times ind2 \otimes ind1,ref1\text{`}man! \otimes ind1] \quad 200b$$

∀ind1.∀ind2.[offers,ref3ˇcarrot×ind2⊗ind1,
ref2ˇdonkey×ind2⊗ind1,refl ˇman!⊗ind1] 200c

Expression b represents the (many) readings where each one of a set of men offers some set of donkeys some set of carrots. Expression c is produced from b by application of index dependency, assuming the upper bounds of the indices are equal. It represents the reading where each one of a set of men offers a set of donkeys its own set of carrots.

13.5.Ordering Index Propagation

I have already mentioned that, in the case of weak indices propagating to weak indices, ordering of the propagations is unimportant. However, now that we have the possibility of more than one index appearing in any quantified expression, we must consider the ordering of the operation manipulating such an expression in full generality. For example, consider sentence 201a, and its propagated translations, 201b and c.

"Every man offers a donkey every carrot." 201a
∀ind1.∀ind2.[offers,ref3ˇcarrot!⊗ind2,ref2ˇdonkey⊗ind1,refl ˇman!⊗ind1] 201b
∀ind1.∀ind2.[offers,ref3ˇcarrot!⊗ind2,ref2ˇdonkey⊗ind2,refl ˇman!⊗ind1] 201c

According to our rules, we must propagate ind1 left, but we must also propagate ind2 right (both subject to the Index Application Rule). However, either one of these precludes the other, because strong indices cannot propagate to strongly indexed references. Nonetheless, we need to consider the readings produced in both cases.

If we perform left propagation on ind1, we get reading b, where each one of a set of men offers some donkey each of a set of carrots. This is a correct reading, characterisable in FOPC as

$$\forall m.man(m) \Rightarrow \exists d.donkey(d) \wedge \forall c.carrot(c) \Rightarrow offers(d, c, m)$$

If we propagate ind2 right, we get reading c, where each one of a set of men offers a particular carrot to a particular donkey. This, too, is correct; the FOPC equivalent is

$$\forall c.carrot(c) \Rightarrow \exists d.donkey(d) \wedge \forall m.man(m) \Rightarrow offers(d, c, m)$$

Note that, if we applied both propagations, we would be representing a reading where each man offers each of a different set of carrots to a different donkey. This would be incorrect. The correct reading where there is only one donkey can be obtained (if

necessary) through partition of entity tokens representing the sets, as specified in Rules 16 and 17.

I conclude, therefore, that we must allow the index propagation operations to be carried out in any order. This accords with the idea, made explicit in the last section for index dependencies, that propagation is an operation between two references, and that propagation in larger lists of references is composed of a sequence of propagations between two references.

Note that if we were relying on quantifier scope shuffling alone in FOPC, we would overgenerate, because the incorrect reading mentioned above would be generated when the existential was inside the scope of both universals.

14. Sentences containing Context Extensions

14.1. Quantifier-Free Context Extensions

By extension of the earlier argument regarding the nature of indirect objects and other predicate modifiers, now that I have dealt with three-place predicates, we must cover context extensions, which are the alternative possibility. This is particularly important, because, in the next section, I will use context extensions to cover dependent reference, like the "donkey existential" in the donkey sentence.

Consider sentence 202a. This example is rather contrived, because more natural examples tend to contain more definite references, which do not exhibit interesting behaviour in this context; nevertheless, it is a well formed sentence. Note the ambiguity between readings, mentioned earlier, where the donkeys are in the paddock and where the seeing (*ie* the man) is in the paddock. The unpropagated translations are shown in 202b and c, respectively.

"A man saw every donkey in a paddock." 202a

∀ind1.[in,ref3⁻paddock,ref2⁻donkey!⊗ind1] ▸
[saw,ref2⁻donkey!⊗ind1,ref1⁻man] 202b

∀ind1.[in,ref3⁻paddock,ref1⁻man] ▸ [saw,ref2⁻donkey!⊗ind1,ref1⁻man] 202c

Consider first the body (to the right of the ▸) of 202b, in isolation.

There is no question of leftward propagation of the index, because there is no reference on the left to which to propagate it. However, since ref1 is indefinite, we must, according to

the index application rule and the definition of strong index propagation, propagate the index right. This leaves us with reading, 203, where there is some number of men, between 1 and the number of donkeys.

$$\forall ind1.[saw,ref2^{-}donkey!\otimes ind1,ref1^{-}man\otimes ind1] \qquad 203$$

Now, take the context extension of the same expression. We propagate leftwards, to get reading 204, where the number of paddocks is between 1 and the number of donkeys.

$$\forall ind1.[in,ref3^{-}paddock\otimes ind1,ref2^{-}donkey!\otimes ind1] \qquad 204$$

The combination of these two ambiguities gives us the multiple readings we need. Figure 17 represents the extremes of these where we have either one or the maximum number of

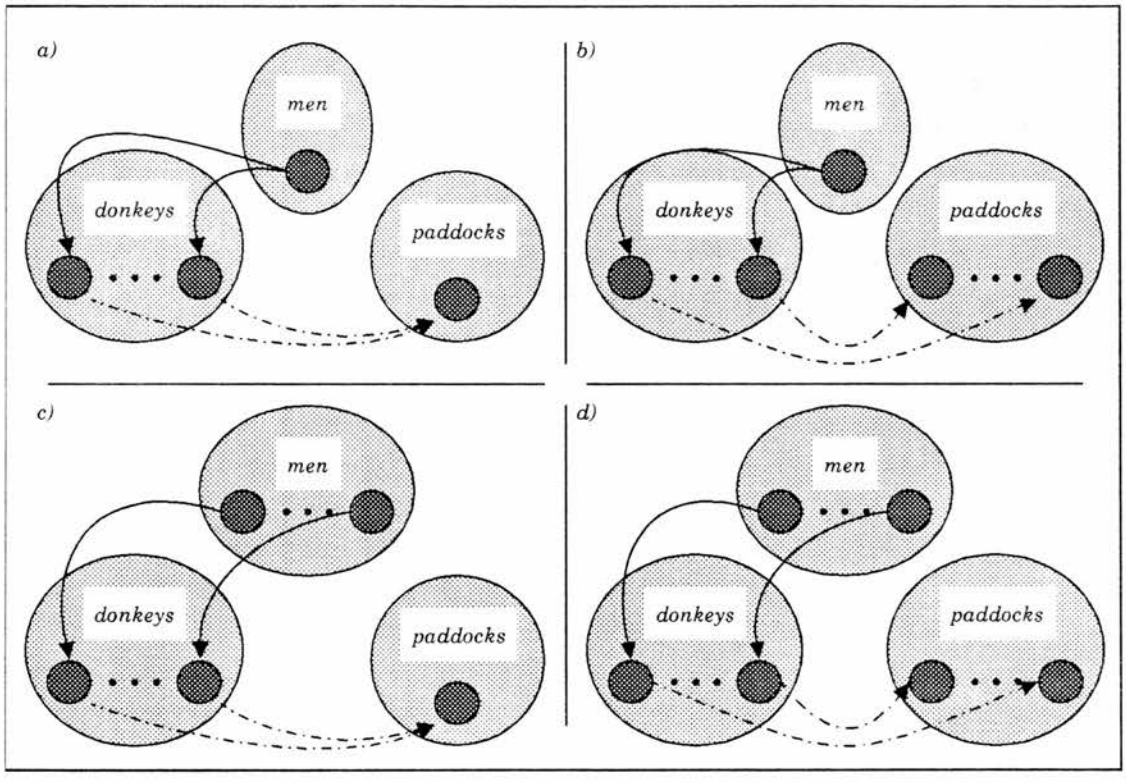


Figure 17: *Extreme readings of "A man saw every donkey in a paddock."*
a) One man saw several donkeys, all in the same paddock.
b) One man saw several donkeys, each in a distinct paddock.
c) Each donkey in the paddock was seen by a distinct man.
d) Each donkey was seen in its own paddock by a distinct man.

donkeys and/or paddocks. Solid arrows denote seeing; broken ones denote location.

Next, consider sentence 202c. In the vast majority of cases, the location of an actor specifies the location of the action taking place – here, I will make this generalisation as a simplifying approximation, so both the man and the seeing are in the paddock. This time,

we cannot consider the body of the expression and the context extension separately, because the two have a reference symbol in common.

Each reference symbol in a GRL expression in George is associated (by binding) with a particular reference in the input. Therefore, it does not make sense for two occurrences of that symbol in a GRL expression to be different, unless one was derived from the other by index partition. (Otherwise, we would be saying that one reference in the input could take two different surface forms at once, which is nonsense. Therefore, the rules of strong and weak index propagation require that if an index propagates to one occurrence of a reference symbol, it propagates to all occurrences, and new quantifiers are inserted if necessary.) Taking the references in the body first, then, we apply rightward propagation of the strong index in the example to give the intermediate expression 205.

$$\forall ind1.[in,ref1^{-}man\otimes ind1,ref3^{-}paddock] \blacktriangleright [saw,ref2^{-}donkey!\otimes ind1,ref1^{-}man\otimes ind1] \quad 205$$

Note, this propagation has affected the occurence of ref1 in the context extension as well as that in the body, as per the definition of index propagation. We now treat this indexed reference in the usual way under index propagation: that is to say, we allow rightward index propagation, giving us the reading in 206.

$$\forall ind1.[in,ref1^{-}man\otimes ind1,ref3^{-}paddock\otimes ind1] \blacktriangleright [saw,ref2^{-}donkey!\otimes ind1,ref1^{-}man\otimes ind1] \quad 206$$

The extremes of the readings denoted by this translation are represented in Figure 18.

14.2. Quantifiers in Context Extensions

Finally, this approach gives us the correct behaviour when the "optional argument" of the predicate (the location in example 202) is quantified. Consider, for example, sentence 207a (which gives us the same attachment ambiguity as before, as shown in 207b and c). Remember that the quantifier covers the whole context extended expression.

"A man saw a donkey in every paddock." 207a

$\forall ind1.[in,ref3^{-}paddock!\otimes ind1,ref2^{-}donkey] \blacktriangleright [saw,ref2^{-}donkey,ref1^{-}man]$ 207b

$\forall ind1.[in,ref3^{-}paddock!\otimes ind1,ref1^{-}man] \blacktriangleright [saw,ref2^{-}donkey,ref1^{-}man]$ 207c

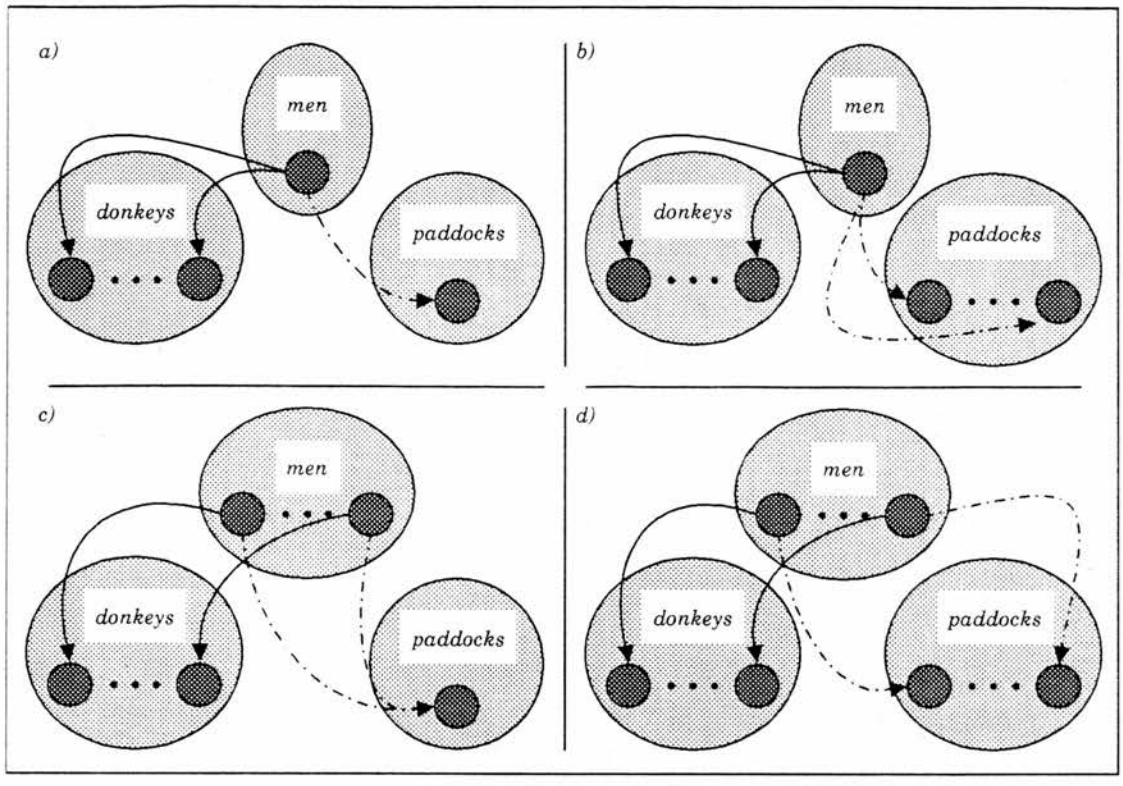


Figure 18: *Extreme readings of "A man saw every donkey in a paddock."*
a) In one particular paddock, one man saw several donkeys.
b) In each of several paddock, one man saw several donkeys.
c) In a particular paddock, each of several men saw a distinct donkey.
d) In each of several paddocks, a distinct man saw a distinct donkey.

Let us take reading 202b first. First, we must propagate the index in the context extension rightwards. When we do so, as before, we must replace the ref2 reference in the body with the new indexed version of ref2. This leaves us with expression 208:

$$\forall ind1.[in,ref3^{-}paddock!\otimes ind1,ref2^{-}donkey\otimes ind1] \blacktriangleright [saw,ref2^{-}donkey\otimes ind1,ref1^{-}man] \qquad 208$$

Now, we must propagate the new index rightwards in the expression body, to give 209:

$$\forall ind1.[in,ref3^{-}paddock!\otimes ind1,ref2^{-}donkey\otimes ind1] \blacktriangleright [saw,ref2^{-}donkey\otimes ind1,ref1^{-}man\otimes ind1] \qquad 209$$

The English gloss of this translation is "A particular man saw a particular donkey in each of several paddocks.". This reading is correct; no others are available.

Now we will take reading 207c. As before, we must propagate the index in the context extension leftwards; as before, it carries through to the body. Since the new index is now

on the rightmost argument of the predicate, we must propagate it leftwards according to the index propagation rule. This leaves us with 210, which is indeed the only correct reading.

"Each of a set of men saw a particular donkey in a particular paddock." 210a

∀ind1.[in,ref3˜paddock!⊗ind1,ref1˜man⊗ind1] ▸

[saw,ref2˜donkey⊗ind1,ref1˜man⊗ind1] 210b

14.3. Sentences containing Dependent References

Finally, we must discuss the behaviour of indices when applied to *dependent references*. Dependent references are (often pronominal) references occurring near (often in the same sentence as) another reference which in some sense determines their meaning. Sentences containing what I have called the "donkey existential" (see Chapter 1) are an example of this problematic kind of sentence – see [Kamp, 1985]. George does not currently have the ability to deal with dependent references in examples where the dependent reference is not in the same sentence as the reference upon which it is dependent. However, the behaviour within sentences is correct; the extension of this behaviour to inter-sentence dependence is beyond the scope of the work presented here, and is therefore deferred to Chapter 9.

14.4. Index Propagation to Dependent References: Donkey Sentences

The essential problems with donkey sentences were discussed in Chapters 1 and 3. They are embodied in the reference of the pronoun in sentence 211a, and in references to the same entity by subsequent referring expressions. In the example, the sort "it" expresses the property of not being a person – as usual, this translation is merely presented as adequate for experimental purposes, and not as necessarily philosophically correct. Expression 211b shows the initial form of the translation, before application of index propagation or expansion.

"Every man who owns a donkey beats it." 211a

∀ind1.[owns,ref2˜donkey,ref1˜man!⊗ind1] ▸

[beats,ref3˜it!,ref1˜man!⊗ind1] 211b

In Kamp's analysis, the first interesting issue is the singular surface form of the referring expressions "a donkey" and "it" – even though these references are apparently singular, there is a readily achieved reading where there is a set of donkeys, and the pronoun refers to each member of the set. This effect is produced by the scope of the quantifier in "Every man". Now, this is exactly the kind of behaviour George is designed to deal with. So how does it deal with donkey references?

As always, the first thing we must do is apply any of the index manipulation rules which are applicable here. Index expansion cannot apply, because the index, ind1, is unbounded. Weak index propagation cannot apply, because there are no weak indices. Strong index propagation, however, is applicable. Applying it, we get just one expression, that shown in 212, the derivation proceeding as follows. The strong index propagation rule (Rule 15) says that ind1 must propagate left in both the body and the context extension subject to the index application rule (Rule 18). Propagation on to the indefinite reference in the context extension is therefore successful. However, the index application rule precludes propagation of an index on to a simple definite reference. Therefore, ind1 cannot propagate left on to ref3.

$$\forall ind1.[owns,ref2\hat{}donkey\otimes ind1,ref1\hat{}man!\otimes ind1] \blacktriangleright [beats,ref3\hat{}it!,ref1\hat{}man!\otimes ind1] \qquad 212$$

If we allow a binding of ref3 to an entity already in the discourse, then this is certainly a correct interpretation of the sentence – consider example 213, where the pronoun refers to an entity introduced earlier:

"In order to reduce cruelty to animals, the town council has bought a whipping horse.
Every man who owns a donkey beats it." 213

However, we also need to produce the alternative reading where there is a set of donkeys in some underspecified relationship with the men (which is already expressed by 212) and the pronoun is specific to each man – *ie* it refers to "the donkey which the man owns". One solution to this might be that we need to treat pronouns differently from other referring expressions here – indeed, in general, this is a common view (*eg* [Webber, 1979]). However, the problem behaviour is also found in the (clumsy, but nonetheless well-formed) sentence:

"Every man who owns a donkey beats the donkey." 214

Therefore, using the pronominal nature of the reference does not solve the problem in general. In fact, we will need to introduce a final new piece of behaviour into the system to deal with such references. The behaviour is described in terms of the combination of quantification and context extension; it produces the required reading for both the pronominal form (211a) and the (non-pronominal) definite noun phrase (214). The full definition is given in Rule 19.

Let E2 ▸ E1 be a context extended expression. Let R1 be a reference in E1, and R2 a reference in E2, both before index propagation. Then:

If the set of entities sort consistent with R1 is a superset of the set of entities sort consistent with R2, R1 may optionally be indexed with any strong index associated with or propagating on to R2. The candidate set of R1 is then reduced to be the same as that of R2.

Rule 19: *Index Propagation through Context Extension*

Now, how does this work? Consider again 211b. Compare the references in the context extension (E2 in the rule) and the body (E1 in the rule). First, the "men" references are identical, so the new rule applied to ref1 is vacuous. The "men" reference in the body cannot corefer with the "donkey" reference (R2 in the rule) in the extension because of both sort and number inconsistency. However, the "it" reference (R1 in the rule) can do so. Therefore, when we follow the strong index propagation rule (Rule 15) and apply ind1 to ref2, we may also optionally apply it to the "it" reference, ref3. The option gives us the required ambiguity between the two readings detailed earlier in this section. Note that we can analyse the translation of sentence 214 in exactly the same way; the nature of the reference, other than that it is consistent (*ie* has the same candidate set) with the appropriate reference in the context extension, is immaterial. Thus, the same reasoning process works for the plural equivalent of 211, shown in 215:

"Every man who owns some donkeys beats them." 215

Note that this rule is not symmetrical – propagation may only take place rightwards: the direction indicated by the context extension operator. This means that we still get the correct (unique) reading for a sentence like 216:

"Every man who owns it beats a donkey." 216

where the pronoun can only refer to an entity introduced previously. If we could pass indices from the body to the context extension (*ie* Rule 19 in reverse), we would be able to infer that "it" was a set reference to the donkeys intoduced in this sentence, which would certainly be incorrect.

For completeness, I must state that no equivalent operation is available for weakly indexed references. The reason for this, fitting with the intuition of our nomenclature, is that the weak quantifiers do not affect each other through the division caused by subordination. Consider example 217:

"Some men who own some donkeys beat them." 217

The point is that in the donkey sentence we have a singular surface form ("every") with a plural semantics; ordinary singular surface forms coreferring with it must therefore be in some sense converted so that they can refer to the set correctly. This is the motivation for the notational division in GRL between number and sort information: the two must often be treated separately. Now, in 217, all the sets are plural already. (This leaves us with more work to do to decide which man owns which donkey(s) and which man beats which donkey(s), as and when necessary. However, I have already shown, in Section 10, how to perform such elaboration.)

This, then, accounts for the first of Kamp's donkey problems: accounting for reference by the singular pronoun to all of the members of the set of donkeys. The second, in George, is solved as a direct result of the solution to the first. The second problem is this. Compare the two discourses 218a and b.

"Every man who owns a donkey beats it.
The donkey is sad." 218a

"Every man who owns a donkey beats it.
The donkeys are sad." 218b

Kamp's suggestion is that the first discourse is referentially ill-formed, while the second is well-formed, because the agreement between the subject of the second sentence must be with the set of donkeys. Now consider the analysis of the two discourses in George. They both start with the translation and bindings shown in 219a and b, assuming the prior existence of a set of men, specified by e1.

$$\forall ind1.[owns,ref2^{\frown}donkey \otimes ind1, ref1^{\frown}man! \otimes ind1] \blacktriangleright [beats,ref3^{\frown}it! \otimes ind1, ref1^{\frown}man! \otimes ind1] \quad 219a$$

{⟨{∀ind1.ref1~man!⊗ind1},{e1~man}⟩,
⟨{∀ind1.ref2~donkey⊗ind1},{e2~donkey}⟩,
⟨{∀ind1.ref3~it!⊗ind1},{e2~donkey}⟩} 219b

Now, consider what happens when we introduce the second sentence of 218a. This gives us the translation in 220.

coref(ref5~sad, ref4~donkey!) 220

Now, we follow the George dereferencing algorithm given in the foregoing sections. First, we know that ref4 and ref5 refer to the same entity, and therefore share the same binding. Second, we look for an entity token in the entity tokens database which is compatible in sort with ref4 and ref5. The only one which is so is e2, which specifies the set of donkeys from the first sentence. However, we have a rule (Rule 4) which states that simple (*ie* singular) references may not be bound to entities appearing in singleton bindings with indexed (*ie* plural) references. Therefore, e2 is ruled out as a candidate, as required.

When we consider the second sentence of 218b, we have the converse situation. e2 is selected by sort consistency as a candidate as before. But the reference to the donkeys is now indexed, so Rule 4 does not rule out the binding. We therefore achieve the required translation, as shown in 221.

∀ind1.coref(ref5~sad, ref4~donkey!⊗ind1) 221a
{⟨{ref5~sad, ∀ind1.ref4~donkey!⊗ind1},{e2~donkey~sad}⟩} 221b

The reason why this works is simple. George changes the representation of sentences like this to fit their correct readings. Thus, when there is a set of donkeys, the set is explicitly represented as such, and the original singular reference from which it was created is discarded. Therefore, there can be no incorrect referent for subsequent references to refer to.

As I said before, it is possible to imagine a solution to the harder, more general problem in example 222, where the quantifying effect of the "Every man" reference passes to the next sentence. This would require a notion of focus – I therefore merely present it here as a possible extension, and will give detail in Chapter 9.

"Every man who owns a donkey beats it.
He often mistreats it in other ways, too." 222

Finally, to show that the behaviour described in this section is not confined to an isolated simple case, consider sentence 223.

"Every man who offers a donkey a carrot gives it to him." 223a

∀ind1.[offers,ref3~carrot,ref2~donkey,ref1~man!⊗ind1] ▸
[gives,ref4~it!,ref5~male!,ref1~man!⊗ind1] 223b

There are four possible readings here, generated by the two-way ambiguity in each of the pronoun references: either, both, or neither of them may refer to entities already existing in the discourse. In the current George implementation, there are no selectional restrictions on predicate arguments – therefore, the example has been contrived so that the pronoun references are not ambiguous in any way other than the way we are studying here. This does not affect the operation of the theory; if the references were ambiguous, more possible readings would be produced.

So, let us apply the rules as before. First, we have the compulsory leftward propagation of the strong index in the context extension to the other arguments. This yields reading 224.

∀ind1.[offers,ref3~carrot⊗ind1,ref2~donkey⊗ind1,ref1~man!⊗ind1] ▸
[gives,ref4~it!,ref5~male!,ref1~man!⊗ind1] 224

Given the definition of strong index partition and expansion, this denotes all the acceptable readings, if, for each of the pronoun references, either there is an existing entity which is a candidate for binding or the corresponding reference in the context extension has not gained an index by propagation. For example, 224 is a correct translation if there is an existing singleton entity token consistent with sort male, to which ref5 can be bound, and either there is another consistent with "it", or ref4 is bound to the same token as ref3.

However, we may also apply the new operation of index propagation through context extensions (Rule 19). This will allow us optionally to propagate ind1 to ref5 or to ref4 or to both, in 224. These readings may also be correct, again depending on the current context. In particular, 224 gives the reading we are most interested in here, where "him" refers to the donkeys, and "it" refers to the carrots.

14.5. Over-Generation from Dependent References

It seems, then, that our extension to index propagation gives us the readings we need. But we must also ask the question: does it give us more than we need? The question is more pertinent here than elsewhere in this discussion, because we are appealing to an idea other than enumeration of sets to give us our readings. Before, generation could only go as far as exhaustive enumeration, which was a desirable extreme. Now, we have the wherewithall to generate references by another process, so we must check that process for over-generation.

First, let us consider when the donkey existential phenomenon appears. Looking at the standard donkey sentence in 211, we can get an immediate picture of the circumstances necessary. First, we must refer to a prototypical entity ("a donkey") inside the scope of a strong quantifier ("Every man"). Then, we must refer to that entity, with surface form syntactic agreement (so it may be an individual or a set) also within the scope of the same strong quantifier.

This analysis of the state of affairs giving rise to the donkey sentence behaviour is actually very constrained. First, we must have two potentially coreferring references in the scope of the same strong quantifier. But referring expressions appearing in the same clause cannot normally corefer (in the general linguistic context, not just in George). Therefore, the donkey sentence must consist of at least two clauses, both of which must be in the scope of the quantifier; the coreferring expressions must be in different clauses. Now, because an entity cannot be referred to until it has been introduced into a discourse, the introductory reference must appear in the surface form before the reference to it. English syntax is such that relatives immediately follow the noun phrase which they modify. Therefore, the introductory reference must be in the relative clause.

Let us now translate this into George terms. We need two clauses in the scope of the same quantifier. The only way of achieving this is through context extension. One of the two ways a context extended expression can be introduced is by translation of a relative clause. If we have an input sentence containing a strong quantifier, then its translation contains a strong index, whose scope is the whole translation. Finally, because the introductory reference in the input is in the relative clause, it appears in the context extension of the GRL translation, as required by Rule 19, defining this new kind of index propagation.

Thus, the concepts defined in George have allowed us to define a single rule (Rule 19) capturing this behaviour exactly. To demonstrate this further, another more obscure example is given below.

Consider sentence 225a, whose initial propagated translation is given in 225b.When we perform this propagation, the rule of referential consistency forces us to rewrite the occurrence of ref1 in the body of the expression as well.

"A man who owns every donkey beats it." 225a

$\forall$ind1.[owns,ref2^{-}donkey!$\otimes$ind1,ref1^{-}man$\otimes$ind1] $\blacktriangleright$

[beats,ref3^{-}it!,ref1^{-}man$\otimes$ind1] 225b

This translation is correct on its own, if and only if there is an entity consistent with "it" already in the discourse. This reading gives us some number of men, each of whom owns a set of donkeys, and each of whom beats some other thing, unspecified in the example above. However, we may now apply Rule 19 to propagate ind1 from ref2 along the context extension to ref3. This leaves us with the final translation, 226.

$\forall$ind1.[owns,ref2^{-}donkey!$\otimes$ind1,ref1^{-}man$\otimes$ind1] $\blacktriangleright$

[beats,ref3^{-}it!$\otimes$ind1,ref1^{-}man$\otimes$ind1] 226

This expression represents the only possible reading of the sentence, other than that given for 225, above. The reference containing ref3 may be bound to some set appearing earlier in the discourse, or to the entity token introduced or referred to by ref2. This gives us the reading where each donkey is owned by some man (maybe the same, maybe different for some or all donkeys), who beats the donkey he owns.

15. Summary

In this chapter, I have presented a new view of quantification in natural language and an associated set of operations which allow the reduction of vagueness in adaptable representations of underspecifed quantified references.

The theory rests on the notion that quantification is a property applied to references, rather than a property of the references themselves. This notion admits direct and explicit representation and manipulation of interactions between quantifiers appearing in the same sentence.

I have presented a first approximation to a view of quantifiers couched in terms of the concept of "strength". Quantifier strength determines the ability of a quantifier to affect the behaviour of other quantifiers, and the nature of any effect it has. I have shown how the notion of quantifier strength may be used to categorise the manipulations, and facilitate their use in deriving more specific referential expressions from vague ones.

I have shown how the adaptable nature of GRL and the George system allow step-by-step manipulations, rendering representations more specific a little at a time. This has allowed the approach to cover extremely ambiguous sentences which would be difficult under conventional approaches based on early enumeration of readings.

16. Afterword

This and the preceding three chapters have presented George's contribution to linguistic theory in isolation. It now remains to place this contribution in context of related work towards the same aims. Such a context, with an overall summary of the work presented in these four chapters, is supplied in the next chapter.

Chapter 8
Discussion and Relation to Existing Work

Abstract

There are four main areas in which the George system and the associated theory extend or improve upon existing work. These are: incremental categorial parsing; incremental analysis of noun phrase reference, and the associated adaptable representations; and reasoning about reference to subsets of underspecified sets.

Each of these is compared with the existing approaches, especially with the work presented in Chapter 2. The notion of adaptability is summarised, and presented as a useful new idea.

1. Introduction

In this chapter, I will compare the theory of the George system, presented in the foregoing five chapters, with the work of those researchers introduced in Chapter 2. I will suggest that the theory presented in this document constitutes progress in automatic parsing, in automatic analysis of reference (particularly in an early/incremental context), and – primarily – in automatic analysis of and reasoning about set reference.

I will also argue that the discussion of representation here explicitly addresses issues recognised as fundamentally important in the work of other researchers in a clearer and more useful way than before.

Recall from Chapter 1 that I aim to substantiate three main claims, which are as follows.

1. We need to be able to analyse, reason about, and represent reference to sets which are subsets of other sets existing in a discourse, but not fully specified therein, in particular in terms of number.

2. We cannot, until the end of an input discourse, fix substitutions for names of discourse world entities for the referring expressions in our input, because it is not possible to be sure that a candidate entity is the correct choice until it is known that no more information is forthcoming. Thus, representations must be *adaptable*.

3. As a corollary of 2, we need to produce *partial* representations in order that we may represent intermediate stages along the path to the eventual analysis.

The organisation of this chapter is such that the issues are presented in order of increasing importance.

2. Incremental Parsing with Adaptable Representations

2.1. Introduction

In Chapter 5, I introduced a new means of parsing natural language using an extension of the Categorial Grammar of [Adjukiewitz, 1935], and presented an implementation, the George Parser. In theory, the George Parser is designed to be strictly incremental – that is to say, it is intended to combine new word meanings with its existing interpretation of the input with no possibility of non-lexical indeterminism – which is made possible by allowing ambiguity in the input to be implicit in the representation which constitutes its output. In fact, however, it has proved necessary to restrict this implicit ambiguity of representation to certain forms of ambiguity only (which, I will argue later in this section, is desirable in the final analysis).

Because the parser is strictly incremental (those readings which are not representable ambiguously being evaluated in parallel), mechanisms are required to replace the operation of the stack used for temporary storage of partial constitutents in (for example) more conventional shift/reduce parsers. Also, the notion of deliberate implicit ambiguity in a representation itself requires the addition of a mechanism for deriving one or more of the possible readings when (if) this becomes possible.

Aside from these two new mechanisms, I must be able to account for the apparent preference for certain readings which [Crain & Steedman, 1986] suggest may best be expressed by parallel elaboration of different possible meanings with a preference attached to certain interpretations. Without such a notion a future extension of the George system will fail to account, for example, for the problem of garden path sentences.

To place the George Parser in context of the chosen existing researchers' work, I will briefly discuss Mellish's DCG-based parser, focussing in more detail on Haddock's deeper writing on the subject of parsing, and on Steedman's Combinatory Grammar which Haddock uses. Webber did not discuss the issue of parsing *per se*, though she specifies a very high level algorithm which might be interpreted as specifying (in Mellish's terms) an

incremental evaluation process taking place only after a complete syntactic analysis ([Webber, 1979, pp 2-61,2-62]).

2.2. Determinism in Parsing

I explained in Chapters 3 and 5 that the motivation for choosing incremental parsing without backtracking for the George Parser is to provide a strict framework for experiment in the incremental evaluation of reference and adaptable representation of discourse. A secondary goal is to improve efficiency by reducing the overhead of maintaining various possible readings during phases of local ambiguity; this can reduce (*eg*) the need to recalculate the reference of referring expressions translated identically in different enclosing parses. Mellish and Haddock have both shown that some considerable degree of early/incremental evaluation can be formally modelled; and strong evidence exists that the human sentence processing mechanism is incremental.

Note, now, that it is far from clear that the parsing in either Mellish's or Haddock's system was in the strong sense incremental. [Haddock, 1989] correctly argues that Mellish's system is certainly not strictly incremental. Mellish's parsing is performed traditionally top-down, with a DCG parser, so application of constraints arising from verb phrases dominating the noun phrases on which he concentrates happens after the noun phrases, and not interleaved with them, as would be the genuinely incremental case; he does not present the parsing in his system as incremental, but just the reference evaluation. This does not detract from the theory, though, because the mechanism is still incremental enough to be interesting, and anyway Mellish intended to be incremental only at the level of noun phrase evaluation, which was a useful first approximation in his seminal work.

[Haddock, 1989] claims to be rather more strongly incremental, to the extent that Haddock conflates Mellish's distinct terms, early and incremental, into just incremental: within a given reading, he parses word by word, which is maximally early in all cases. However, for purposes of experiment, Haddock uses a backtracking shift/reduce parser, and sacrifices many of the potential gains of incremental parsing to a desire for simplicity. He does, however, devote much attention to issues of parsing, proposing a chart parser as a good solution to the incremental parsing problem.

I have argued, as did Haddock, that an obvious choice of grammar for such a parser was an enchanced Categorial Grammar. This brought with it some problems which I addressed in an unconventional way. In particular, the issues of stacking of partial results, of spurious ambiguity, and of searching between different syntactic parses were raised. I proposed the

solutions of protraction and coercion to cover the first and last of these. By making possible and enforcing strictly incremental parsing, one *a priori* removes the so-called *spurious ambiguity*, which can only arise when words in a sentence may be combined in more than one order.

At the coarsest level, the George Parser has two mechanisms for dealing with form-class ambiguity: either simple transference of the ambiguity into the output representation, or forking of the entire parsing process. I have not attempted to characterise in general which option is taken when, but there are certain broad statements one can make. For example, in the case of a syntactic constituent which appears optionally in a sentence the rest of whose structure is unaffected either way (*eg* a relative clause), we can represent the sentence assuming it will be present, and then remove it when it proves to be absent (or, arguably, the converse). This is the approach taken in Chapter 6 to noun phrase post-modifiers. More generally, a line can be drawn between syntactic forms which (in George) are easily conflatable (like np and np/nmod) and those which are not (like passive participle and active verb, as in the garden path sentence, 227). Of course, the ease with which representations can be conflated is entirely dependent on the choice of representation language.

"The horse raced past the barn fell." 227

2.3. Adaptability and Ambiguity

In Chapter 5, I explained that, in George, there is a distinction between ambiguous representation and multiple translations proceeding in parallel. Thus, George's position on adaptability is, because of practical factors, not absolute; not all ambiguous input is translated as ambiguous GRL.

Now, it would seem that in a completely adaptable representation system which always maintained ambiguity in full it would be impossible to simulate effects (like garden pathing) which arise from preferences for certain readings, because there are no distinct translations to which preferences may be attached. If this is so, I cannot claim that such a system is a good emulation of human behaviour.

If we follow [Crain & Steedman, 1986]'s line and attempt to make judgements of preference between possible readings composed in parallel, we can account for the garden path effect in 227 even in the partially adaptable framework I have presented here, as follows. There are two relevant lexical entries in George for "raced", because the two

translations are sufficiently different not to be conveniently representable in the same ambiguous form. Therefore, after the head noun phrase, the parse forks and each translation of "raced" is incorporated into a different possible partial translation of the whole.

Crain and Steedman suggest that we should attach some likelihood value to these two readings, which will presumably be based on the word "raced" (since it is the pivot, as it were, between the possible readings). In this example, the active reading is preferred over the passive.

When we reach the closure of the preferred reading, after "barn", we may assume that we have found a correct reading; we might therefore dispose completely of the other reading (which is in fact the correct one). Thus, when we attempt to incorporate the word "fell", we no longer have an appropriate partial sentence into which to incorporate it, so the parse fails, simulating the most extreme garden path behaviour in humans. Different, weaker ways of disposing of the less favoured reading can simulate different degrees of garden-pathed-ness. This kind of approach would be much harder (or maybe even impossible?) if we insisted on ambiguous representation for all ambiguities.

Thus, George's use of adaptable representations does not, as one might expect, remove the option of preference between readings; and the above argument lends weight to the suggestion that while deliberate ambiguity (and hence adaptability) is desirable, there are limits beyond which it becomes counter-productive. Perhaps, then, the garden-path (or not) nature of an ambiguous sentence can be a guide as to the better way to deal with any ambiguity in the words it contains; and what seemed in abstract to be a limitation in the theory is maybe in reality a positive feature of the system.

While the issue of implementing such a facility is left for further work, it is important to state that the issue has been considered. This psycho-linguistically motivated division between explicit and implicit representation of ambiguity may well coincide with a similar practically motivated one, depending on the choice of representation.

2.4. Removing the Need for a Stack – Protraction

What is the issue about stacks for temporary storage (*eg* storing a head noun phrase while its verb phrase is being parsed) in incremental analysis? Quite simply, it is that I wish to have a single (partial) representation of the discourse we have analysed so far after the incorporation of each new word. Only then can I treat a partial representation in the same

way as a complete one for the purposes of manipulation and inference; if I cannot do this, I cannot easily claim that my representation is adaptable in any strong (and therefore interesting) sense.

Recall (from Chapter 5) that the main reason this is a problem at all is the entrenchment of the λ-notation for semantic functions. Because we generally wish our semantic functions to correspond structurally with our syntactic categories (Montague's "Rule-to-Rule" hypothesis), we therefore impose a directional nature on syntactic, as well as on semantic, combination. In [Lambek, 1957]'s linguistic calculus, syntactic combination was bi-directional with no imposed order, and so the problem did not arise – but then, Lambek gave no semantic analysis. Given the widespread use of the λ-notation, and the accepted modern direction-ordered style of categorial grammar, there seems little choice but to work around the problem, rather than confront it directly.

The main existing solutions to this are the use of a shift reduce parser (in the strong sense, where we may leave constituents on the stack while composing others) and the idea of type-raising (which I introduced, with its associated problems, in Chapter 5) and the related generalised composition.

The point is, though, that any approach we take to this problem will have somehow to encode the behaviour of a stack. Type-raising does so by introducing an extra λ-abstraction. Steedman's Composition and Generalised Composition operations (see [Haddock, 1989]) do so by higher-order manipulation of syntactic categories, again introducing a delay in evaluation by manipulation of λ-terms.

In Chapter 5, I proposed an alternative mechanism for this purpose: *protraction*. I claim that this is a better view than all the above for the following reasons. First, protraction allows us to build a single partial representation for any left-complete sub-sentence, unlike a stack based parser. Second, it is a fully general mechanism usable anywhere in a parse, unlike type-raising, which is applicable *ad hoc* at certain points only (*eg* subject noun phrase, nominal); and its use does not significantly non-determinism, because it is only applied on demand, unlike type-raising. Finally, it allows us to do away altogether with the idea of function composition in categorial grammar, making the grammar simpler and more efficient to use for parsing; in particular, generalised composition, introduced to allow treatment of right-extraction of arbitrary syntactic components, is rendered unnecessary, and thus the problems it introduces (see [Haddock, 1989, pp60,121-123]) are discharged.

2.5. Supporting Ambiguous Representations – Coercion

It seems, then, that a primary potential gain from the use of ambiguous adaptable (partial) representations is an increase in effective determinism in a parser, simply because fewer distinct possible translations need to be considered. However, in order to allow the disambiguation of these representations if and when it becomes possible, we must introduce a mechanism, beyond that which performs straightforward combination of existing partial representations and fresh lexical items arising from new input. This new mechanism must *adapt* the ambiguous representation in such a way that it becomes more specific. However, if we do this in an unprincipled way we run the risk of reintroducing and even increasing the indeterminism which we wanted to remove in the first place.

One good solution to this – corresponding closely with the incorporation of protraction into the grammar's combination rules, and the associated improvement over arbitrary type-raising of lexical items – is to introduce the mechanism in such a way that it is only ever called when it is known to be needed. This embodies exactly the idea that disambiguating adaptations should be made only when the correct disambiguation is known. In the George Parser, because the disambiguating mechanism, *coercion*, is only ever called when we attempt to combine two items, the form of the second can determine how the first is coerced (and the second is never coerced because it is a lexical entry – see Chapter 5).

2.6. Summary

In this section I have argued the following claims about how the use of ambiguous adaptable representations and strictly incremental analysis can improve over existing parsing techniques.

1. (Section 2.2) Adaptable ambiguous representations can increase determinism in parsing and thus improve efficiency over any kind of parser which represents ambiguity explicitly, particularly when semantics and reference are evaluated in parallel with syntax.

2. (Section 2.2) Strictly incremental parsing removes the spurious ambiguity inherent in categorial grammars parsed in other ways.

3. (Section 2.3) The incomplete adaptability of GRL and the George system, and its need to work out some ambiguities as distinct possible translations need not detract from

the elegance and utility of the system. Indeed, there is probably a point beyond which it is linguistically and philosophically appropriate to represent ambiguity explicitly, in order that preferences between different readings may be expressed; George would be able to take advantage of this without major change.

4. (Section 2.4) Two unsatisfactory mechanisms in categorial grammar, generalised composition and the *ad hoc* type-raising, may be rendered unnecessary by the addition of a single uniform mechanism, which I have called "protraction". Protraction can improve both parsing efficiency and clarity of grammar specification, because its use simplifies the set of categorial combination rules.

5. (Section 2.5) Adaptable ambiguous representations require some means of disambiguation. Disambiguation should only occur when there is explicit justification for it. Coercion is one means of such controlled disambiguation.

3. Adaptability in Evaluation of Reference

3.1. Introduction

In the last section, I justified the use of a loose notion of adaptability from the point of view of efficient parsing. Now I will consider adaptability with a more specific slant towards evaluation of reference.

[Mellish, 1981] uses the terms *early* and *incremental* to describe the evaluation of reference in the MECHO system. *Early* refers to the addition of information into the representation, and inference of its consequences, sooner after it appears at the input than in more conventional systems, in particular before the end of the sentence. *Incremental* means that the representation and/or the evaluation process must be geared to allow the repeated addition of small pieces of information (from individual words or phrases), rather than the incorporation of one large translation (from, say, a complete sentence). If a system uses early evaluation then there is a requirement that it be in some sense incremental. This is the reason that [Winograd, 1972]'s SHRDLU fails to cope with underspecified definite singulars – its evaluation is early, but not incremental.

[Haddock, 1989] takes a rather stronger position than Mellish in that his incremental evaluation is at the level of words within noun phrases, rather than of noun phrases within sentences. He then uses the word *incremental* in such a way as to mean "*early* and *incremental*" in Mellish's terms. Significantly, Haddock claims that a cut-off point can be

placed on the evaluation of definite noun phrase reference, so the incremental nature of his system is deliberately limited – especially since he only considers singular definite noun phrases.

One major claim of this thesis is that *incremental* evaluation and/or representation of reference alone are/is not enough. In Chapter 1, I defined the term *strictly incremental parsing* to mean parsing where word meanings are incorporated into a single partial translation precisely when they appear at the input to a parser, and the term *adaptable representation* to mean a representation which implicitly includes ambiguity and which may be manipulated in certain well-defined ways to take advantage of the fact while representing the semantics and reference of partial or complete discourses. How, then, does this adaptability idea differ from the existing notions?

3.2. Adaptable *vs* Early/Incremental

It is mainly the idea of deliberately implicitly representing non-referential ambiguity which makes the difference between Mellish's and Haddock's incremental evaluation and my evaluation of adaptable representations. It is certainly the case that the candidate sets of MECHO and of Haddock's system can represent referential ambiguity, by having more than one member, when the associated referring expression is singular, for example. However, Mellish and Haddock represent syntactic ambiguity in general explicitly by backtracking, there is no question of adaptable implicit representation of ambiguity at any other level. As I said before, adaptability in some degree is required by any early evaluation system, in order that information may be added as parsing proceeds.

Both Mellish and Haddock consider only discourses where "new" information is added to existing discourse entities, and "given" information serves to select between (sets of) them. Because both researchers use explicit known properties (rather than non-contradiction, as explained in Chapter 2) to select candidates, a candidate for reference cannot be ruled out by addition of information to its own definition, as opposed to addition to the referring expression in whose candidate set it appears. This is because a candidate is always known to have the required properties when it is first added to the candidate set. Thus it is nonsensical to add to the candidate's definition a feature which contradicts some assumption made when including the candidate in the first place, because such a feature would be inconsistent with the existing information in the referring expression.

In George, there is no distinction between new and given information: instead, there is the notion of introduction (or not) of entities by introductory (or non-introductory) references

and (crudely) of definiteness in a reference. Any sort and number information is merely incorporated uniformly, whether or not it is new, as long as it does not contradict information already known.

In other words, in Mellish's and Haddock's incremental evaluation, we can always add previously unknown information; but that information can only constrain candidate sets. In an adaptable representation (in the widest sense), we can do more: we can change from a more ambiguous representation to a less ambiguous one, or, at least in principle, *vice versa*; we can pull individuals out of candidate sets by the direct addition of new information to their definitions. In the strongest case, we can in principle change the form of a partial translation altogether, so long as we have a linguistic justification for doing so, arising from the subsequent word. This difference is suggested by the nomenclature: *incremental* connotes "adding to" or "increasing"; *adaptable* connotes an ability to change more freely. Note, though, that George's adaptability always procedes towards more specific readings.

One implication of all this is that a generally adaptable representation requires rather more structure than Mellish's and Haddock's constraint networks – if we are to adapt representations, we need to know exactly what information came from where; in particular, it is often the case that verbs carry new information, while definite noun phrases usually carry given information. Even this basic information is discarded in the constraint network approaches. I will return to this issue in Section 4.

Adaptability of representation is particularly important in analysing set reference. As I showed in Chapter 7, cases can very easily arise where it is necessary continually to re-evaluate and make more specific mappings between elements of sets. Given this, we need to be able to make representations carrying in some sense a large amount of information – all the possible readings of a relation between sets – and then prune this down, not as a process of candidate set refinement, but by adapting the representation to a more specific reading. It is not clear how this would fit in with purely incremental evaluation with candidate sets, which does not address this problem of the exact mapping of underspecified relations. I will return to this issue in Section 5.

3.3. The Extent of Adaptable (or Incremental) Evaluation

One point which I must cover here is the question of when to stop being adaptable. I suggest that, in fact, there is in general no point at which one can say that all the reference in a discourse has been analysed correctly. This is also Mellish's view: although

he would like candidates for definite singulars to be unique, he neither enforces the requirement, nor uses it as a well-formedness judgement. Haddock, however, takes a different view, which is worth pursuing here.

A perennial issue in the discussion of noun phrase references is the notion that the referent of a singular definite referring expression must be in some sense unique – or at least that the candidate set of the reference must be singleton if the reference analysis is to be said to be successful. Preoccupation with this notion led [Winograd, 1971] to specify inadequate analysis routines in SHRDLU (his requirement being that, if a unique referent could not be found at the end of the noun phrase, the analysis failed and flagged an error), and to a serious problem in the generality of Haddock's mechanism, which is exacerbated by Haddock's use of uniqueness of definite referents to select between different syntactic analyses. Haddock himself gives a clear description of this uniqueness problem with SHRDLU (see [Haddock, 1989] pp129-130) – there seems little point in repeating this here – and raises a problem example of his own ([Haddock, 1989], p158).

Haddock's proposal is that

> "a definite NP should refer uniquely by the time it is syntactically closed".

He demonstrates how this rule can be used to deal with sentence 228 in various contexts ([Haddock, 1989] pp132-137).

"The woman saw the boy with the telescope." 228

The idea is that the uniqueness of definite reference, expressed as a *constraint* on the syntactic *closure* of a noun phrase (or *closure constraint*), can select between readings obtained through different syntactic analyses. In particular, consider Haddock's two possible analyses of 228 where a) the prepositional phrase modifies the main verb; and b) the prepositional phrase modifies the object noun phrase. In a context where there is a non-singleton set of boys exactly one of whom has a telescope, only b) is felicitous; where there is only one boy, the prepositional phrase is read as given because Haddock's system does not allow new information in definite references. In the former context, reading a) is ruled out by the definite closure constraint – by the time the closure is reached, after reading "the boy", the candidate set of that noun phrase is not singleton, and so the analysis fails, leaving the syntactic alternative as required. This technique also applies to the disambiguation of the restrictive or non-restrictive nature of the pp modifier.

Haddock claims that his implementation is an embodiment of [Altmann, 1986]'s Principle of Referential Failure (see Chapter 2, Section 3), because "an analysis which treats

subsequent material as [modifying] will be favoured over one which does not". While it is certainly true that Haddock's system conforms with that principle, the converse is not the case – in Haddock's system, only an analysis which treats immediately subsequent material as modifying is favoured over one which does not. If the modifying material is outside the noun phrase in which the disambiguee phrase is embedded, Haddock's theory labels it ill-formed. To see how this can be a serious failing consider the following discourse. The points at which Haddock's theory would reject it as an ill-formed discourse are marked †.

"A farmer owned two donkeys.
He kept one of the donkeys in a field, and the other in his farmyard.
He made Jim, who is a stable boy, get really filthy the other day.
He made the poor boy† fetch the old donkey†, which he had decided to sell†.
Jim got so dirty because it was raining really hard.
The old donkey† was kept in the muddy field†." 229

First, using Haddock's own candidacy rule, that each property required by a referring expression must be explicitly true of each candidate, the parse fails at the introduction of "poor" in the third sentence, because no known entity is poor (*ie* the "poor" information is not given). If we were to use the more liberal non-contradictory rule (used in George, and in [Charniak, 1972]), the parse fails at end of the next noun phrase, because there are two indistinguishable possible referents. Similar effects occur throughout the discourse – note in particular the failure of both readings of the fourth sentence, the non-restrictive reading after "donkey", and the restrictive after "sell".

The problem for Haddock's theory, in this example, is that the disambiguation appears two sentences after the ambiguous reference, and not as part of the same noun phrase. To allow Haddock's theory to admit this discourse, we would need a rather exotic syntactic analysis.

Now, I suggest that in fact this is a perfectly well-formed discourse, if a little convoluted and clumsy. Further, it seems that vagueness of this kind is widespread in natural language. Therefore, inability to analyse such vague reference, or worse, principled rejection of it, constitutes a serious flaw in Haddock's theory.

Having said this, one can certainly imagine improvements which might alleviate the problem to some considerable extent (or even completely). For example, by making the uniqueness constraint enforcible (perhaps by simply throwing away some candidate(s)), and by specifying a moderately intelligent means of reasoning about the enforcement, one

could account for the discourse in 229, though it is not immediately clear how general is the scope of this proposal.

Haddock himself acknowledges that there are problems with the theory, and suggests that Mellish's more liberal approach is maybe more appropriate. Indeed, the discourse above seems to suggest that there is very little mileage in Haddock's definite closure rule at all.

In the George system, I have followed Mellish's assumption that it is never possible to state that a referential analysis is correct. It would seem hard to make a proof of this view, but there is plenty of evidence that making fixed assumptions is dangerous. One extreme example is raised by [Hirst, 1981]:

> "... in the novel *Even cowgirls get the blues* [Tom Robbins, Bantam, 1977] the character named The Countess is introduced on page 63. It is not until page 66 that we find out that The Countess is male, and we are told this only implicitly by the author's referring to him by the pronoun *he* when there is no other possible referent. A human reader is momentarily fazed by this, but finds recovery easy."

A good approach, therefore, is to remain uncommitted on reference, but to assume that the analysis one has is correct in the absence of evidence to the contrary. So, at any stage in analysing a discourse, we need to be able to fix and extract the intermediate truth functional content of an adaptable representation (at least of complete sentences), but also to sustain that representation in order that we may use it to apply and control adaptation if necessary in future. This is the approach used in the George system.

What is more, this approach implicitly agrees with Altmann's Principle of Referential Failure: any new information appearing after an ambiguous reference is viewed as potentially disambiguating that or any other reference (though most of it will have nothing to do with that reference); if the information is not new, then it simply causes no change in the system. And the extra restriction of locality imposed by Haddock's rule is removed, so, if we use a candidate selection rule based on non-contradiction (as is the case in George), the discourse in 229 can be analysed correctly.

Finally, we can deal with the distinction between restrictive and non-restrictive modifiers in the degenerate way described in Chapter 6, so we do not lose anything in that respect by discarding Haddock's definite closure rule.

3.4. Mutually Dependent References

Having claimed that Haddock's rule is incorrect, I must now answer the problem which it was designed primarily to solve, that of references like those in 230.

[Haddock, 1989]	"The rabbit in the hat"	230a
[Winograd, 1971]	"Put the block in the box on the table"	230b

The issue here is that there can be contexts where the embedded simple noun phrases in the examples do not refer uniquely, but the enclosing complex phrases do. Why is this a problem? For both Winograd and Haddock, the issue is an issue at all because of their rejection of ambiguity (and therefore adaptability) beyond the syntactic closure of a noun phrase; thus, in both cases, the theories are able to solve this referential problem in spite of themselves, rather than by their particular merit.

Suppose a context for 230a where there are two hats and two rabbits, and one of the rabbits is in one of the hats. If we take the more relaxed view of definite reference in MECHO and George, the find that, as we parse the example, we arrive at a constraint involving one of a set of two rabbits, and one of a set of two hats. It is in the nature of this kind of constraint (called, in George, a *context extension*) that the entities taking part in it are mutually dependent. The incorrect choices can be ruled out, because they do not take part in such a relationship. This approach was covered in Chapter 6. In fact, we could go further, and rule out the other rabbit as soon as we read the word "in": we know that that rabbit is not in anything. I will expand on this in Chapter 9.

The same approach can be used to disambiguate the attachment ambiguity in 230b, based on the situation modelled in the representation of the discourse world; although it should be noted that here is another case where we might wish to introduce preference. The obvious cases are where there is a block in a box, which must be put on a table; and where there is a block which must be put in a box which is already on a table. The less obvious case is where "the box" is introductory; the block is to be put in a newly introduced box which is on the table. George's view of these three is that one of the first two is available immediately, depending on referential consistency. The third is derived only when the other two fail, by creation of a new entity token to fill an empty candidate set, which is a linguistically defensible position. I gave a fuller explanation of this class of example in Chapter 6, Section 5.4.4.

3.5. Summary

In this section I have argued the following points.

1. (Section 3.2) The early, incremental analysis of reference requires adaptability of representation. Otherwise, it would be possible to represent neither new information, as it appears incrementally in the system, nor its effects.

2. (Section 3.2) The existing incremental approaches cover only the incremental analysis of noun phrase reference, whereas adaptable representations have other uses as well. In particular, representation of and reasoning about predications between individuals within underspecified sets does not fit easily into Mellish's incremental framework.

3. (Section 3.3) There can be no safe limit to adaptability, within a given discourse. It is always possible that hypothesised referents for referring expressions will prove incorrect at a later stage in the discourse. In particular, the need for adaptability can easily extend beyond the end of the referring expression.

4. (Section 3.4) Adaptability of representation need not compromise our ability to analyse problem references of the kind explored by [Haddock, 1989].

In Section 5, I will give more specific details regarding the evaluation of reference to (underspecified) sets.

4. Representation of Semantics and Reference

4.1. Introduction

So far, I have argued on a very general level why adaptable representations are useful in parsing and in basic noun phrase reference evaluation. In this section, I will pin these points down rather more firmly, and, in particular, justify in more detail the choice of representation for the George system.

4.2. Modelling Discourses and Situations

Before we can make any attempt to represent events or situations described in a discourse on a computer, we must have a clear and correct view of what it is we are trying to represent. I make this point because it is far from clear that such a view is always found. In particular, it is often the case that researchers blur the distinction between, on one hand, the world model and the relationships between entities in it and, on the other, the

relationships specified between the referents of referring expressions in the discourse itself.

For example, in his very useful survey of the work on anaphora current in 1976-81, Graeme Hirst [Hirst, 1981] seems to exhibit a particular lack of precision which is widespread. First he defines that

> "ANAPHORA is the device of making in discourse an ABBREVIATED reference to some entity (or entities) ... The reference is called an ANAPHOR and the entity to which it refers is its REFERENT or ANTECEDENT."

In the very next sentence, he then claims that

> "A reference and its referent are said to be CO-REFERENTIAL."

To paraphrase: a reference and the entity it refers to are co-referential, which is not the case by definition. A reader would probably know what Hirst meant, but I suggest that this (and other examples in Hirst's and others' writing) indicates that there is a certain, widespread lack of distinction, at least in thought, between the phrase which introduces an entity (which is what is really coreferential with the anaphor) and that entity itself. Such a theoretical distinction (ideally made explicit) is fundamental to the notion of availability for reference which Webber highlighted in her thesis (which I outlined in Chapter 2, Section 4.2). It is also fundmental to adaptability.

4.3. Amalgamation

[Webber, 1979] makes a compelling argument for an explicit separation between entities in the world model and reference to them in a discourse. Nevertheless, the entities in Webber's world are denoted by tokens which may themselves appear in the representation language (which Webber thinks of as a logic, though no proof of its status as such is given). This *amalgamation* (or mixture) of names (*qua* referring expressions) and objects (*qua* discourse entities) is similar to that found in (*eg*) [Kim & Kowalski, 1990]'s work on meta-logical reasoning. Many logicians (*eg* [Burt *et al*, 1990]) feel that it is fundamentally undesirable in a logic because of the issues of reflection it raises – there is always the question of whether one is making predications of an object or of its name. In particular, if we simply replace a name by the object it refers to, we have fundamentally changed the nature of our representation, in an unmotivated way.

Webber proposes a framework where entities are described and references are made to them as two distinct but related processes. She writes:

> "Discourse model synthesis and anaphor resolution are complementary processes."

The *invoking descriptions* (in Webber's terminology) are maintained as logical formulæ along with the associated discourse, and linked to it by the naming predicate **evoke**, so an entity is described in terms of its properties and of the fact that it is the one referred to in certain sentences in the discourse. Webber describes reference analysis as a process of refinement of the sort of the definitely (ι) quantified variables appearing in her translation of the discourse, and their eventual replacement by the entities which they denote.

An example of Webber's notation and analysis is given in [Webber, 1979, pp 2–64-2–67]. This example is particularly relevant here, because it includes the notion of vagueness in definite reference. The example discourse, viewed as a continuation of an existing discourse, is as follows:

"Bruce found a banana.
It belonged to a woman he knew.
Bruce remembered that the banana had been stolen
from her Monday by a marauding monkey." 231

After analysing the first sentence Webber arrives at the translations shown in 232a and b (where "Bruce" is introductory or anaphoric, respectively; in the former case, the description of e_{43} is created by this sentence):

S_{92} (∃x:Banana) . Found Bruce, x 232a

S_{92} (∃x:Banana) . Found e_{43}, x 232b

e_{43} Bruce

e_{44} ιx: Banana x & Found e_{43},x & **evoke** S_{92}, x 232c

When Webber analyses the next sentence, she first produces a representation like this:

S_{93} (∃x:λ(u:woman)[Knew HE,u]) . Belonged IT, x 233

Next, she converts this "Level 1" (surface form) representation to a "Level 2" (reference-analysed) form where

> "its pronouns are resolved (or at least a bound variable or parameterized individual interpretation ruled out)."

First, assuming that the resolution is successful, with HE referring to e_{43} and IT referring to e_{44}, she produces this representation:

$$S_{93}\ (\exists x{:}\ \lambda(u{:}\text{Woman})[\text{Knew PRO}=e_{43},u])\ .\ \text{Belonged PRO}=e_{44},x \qquad 234$$

(Note here that Webber acknowledges the need to maintain a representation of the pronominal nature of the anaphors, but does so by this extra-logical syntactic hack.)

Webber's rule for extracting descriptions of new entities then gives the following identifying description for the woman:

$$e_{46}\ \iota x{:}\ \lambda(u{:}\text{Woman})[\text{Knew PRO}=e_{43},u]\ x\ \&\ \text{Belonged PRO}=e_{44},x\ \&\ \textbf{evoke}\ S_{93},\ x \qquad 235$$

The translation is now complete. This analysis leaves one question unanswered: why is the entity name associated with the woman, e_{46}, not substituted into S_{93}, and similarly for the banana and Bruce in S_{92}? It is still far from clear that this sentence contains an explicit existential, so why should one appear in a logical translation, especially when there is a viable alternative?

The alternative, where the two entities are not resolvable runs as follows. The Level 2 representation includes explicit pronouns as before, but they are associated with "unknown" discourse entities, arbitrarily labelled "?", thus:

$$S_{93}\ (\exists x{:}\lambda(u{:}\text{Woman})[\text{Knew HE}=e_{?1},u])\ .\ \text{Belonged IT}=e_{?2},x \qquad 236$$

The rule for extracting descriptions then yields this description of the woman:

$$e_{46}\ \iota x{:}\ \lambda(u{:}\text{Woman})[\text{Knew HE}=e_{?1},u]\ x\ \&\ \text{Belonged IT}=e_{?2},x \qquad 237$$

However, Webber makes no attempt to suggest a semantics for these "vague" entities, simply commenting that they may be replaced by fully specified ones when this becomes possible. There is a weak correspondence between these and Mellish's *reference entities*, but, because of the lack of a formal definition, it is hard to say what they really mean. What seems certain is that they perform very little function at all in the system, other than enabling the early construction of an identifying description. Regrettably they are not associated with the notion of a candidate set, which would have been a good solution to this problem. To be fair, I should mention that there is a fairly strong flavour of adaptability about this feature of Webber's system; even though she seems to view it as a remedial hack, rather than a feature.

The point, at a more general level, is this. One must have some mechanism performing the same function of Webber's substitution and **evoke**, or one will never be able to make connections between the entities and the (referring expressions in the) discourse. However, the necessity of amalgamation within a representation is questionable, especially since it actually makes more work for the analyser, rather than less. In particular, each time a variable is replaced by an entity, or the description of a new entity is detected, the reference of all the variables must be re-evaluated, which is a non-trivial task involving search through the entire discourse representation. This would not be the case if the evaluation were treated as constraint network satisfaction, as in MECHO; and the constraint network approach requires that the representation not be amalgamated (at least in this sense). An alternative view of this approach is to view the entities themselves as (in some sense) *arbitrary objects* (see [Fine, 1985]) whose specifications become more precise as the discourse proceeds, or as though they were merely pieces of discourse with associated candidate sets (as in George). In particular, these roughly equivalent ideas admit early (partial) evaluation of reference even in ambiguous cases, which is not possible in Webber's framework.

What is far worse is that Webber justifies the above use of the **evoke** predicate by drawing a comparison with an explicit reference to a discourse like "You know the man I was talking about yesterday...", and suggesting that one could use it directly in the representation of such an utterance. This seems to be a direct contradiction of her first "strong claim" to represent linguistic structure separately from reference information. In doing so, she seems to show a complete lack of awareness of the amalgamation issue.

That same issue appears again, in a different sense, in Mellish's program; some of the output terms specify situation relations in the discourse world (*eg* "supports"), which do not define the nature of entities, some specify sorting information (*eg* "isa"), which do define entity nature, and some specify naming relations (*eg* "hasname", which indicates what words were used to refer to which entities). Now, the confusion we saw (above) in Webber's system over the status of of the naming relation **evoke** does not appear in the same way in Mellish's work, because his output does not represent the structure of the discourse in the same strong sense. Thus, "hasname(particle, particle1)" is merely one of a collection of pieces of information about particle1 (*viz* that it was referred to as a "particle") and there is no strong claim about translations of virtual utterances like "the particle I was talking about in the last sentence".

Nevertheless, this issue of correct division, both logical and practical, between references in a discourse and entities in a discourse world still arises. In particular, in Mellish's

work, it appears in the distinction between the treatments of indefinite and definite noun phrases, and seems very similar to the confusion mentioned earlier in [Hirst, 1981].

Mellish's program admits the notion of the discourse entity, and also the notion of the explicit reference, so it is possible to write terms like "on(ref(1),ref(2))" which denotes the "on" relationship between two definite reference entities (*sic* – as opposed to the relationship between their referents); the information contained in those noun phrases is expressed by other predications of the ref symbols. What is maybe less straightforward is the idea that a reference entity (which seems to be used to mean "the meaning of a referring expression") is an entity in the discourse world – *eg* [Mellish, 1985], p43:

> "... semantic routines are to be able to handle unevaluated references in the same way as other entities in the world model ...".

Like Webber, Mellish goes on to make various points about representation of linguistic structure; for example, he too points out the need for explicit representation of the definite or indefinite nature of a noun phrase. In MECHO, this difference is notated with reference symbols, "ref(N)" where N is a unique integer, for definites, and with tokens like "particle1" for indefinite entities (*sic*). Note, however, that the status of discourse entities has been subtly changed. Unlike Webber's entities, which are abstracted, independent of the language, and to which we may assign certain properties, these entities are grounded much more firmly in the language itself, to the extent of viewing them as arising directly in the discourse, so an indefinite noun phrase denotes an indefinite entity, as above. Thus, we have indefinite entities and set entities, which may appear in the candidate sets of reference entities, which seem to be notionally on the same level. Thus, while there is still a clear attempt to separate out the meaning of a discourse from the objects discussed in it, that attempt partly fails, because either the different levels are not specified clearly enough (so reference and indefinite entities really are different classes of objects) or because the confusion in [Hirst, 1981] is again present, and the entity introduced by an introductory phrase is being confused (in quite a subtle sense) with the phrase itself.

Also, we see another more serious example of amalgamated representation in Mellish's denotation of sets. A set entity (much like a reference entity) is usually introduced by a plural noun phrase. There are, however, instances of set references which have singular surface form (*eg* "Each" phrases). These must be marked in some way, because subsequent reference to them must agree with the surface form. Mellish uses a notation where the set entities in the world model acquire this linguistic surface form marking, almost as though the description of an entity affected its real world nature. This and the other points above lead me to suggest that this confusion between discourse and discourse world is

potentially considerably more dangerous and insidious than in the work of Webber, because it is implicit in the notation and terminology, rather than arising from deliberate (mis)use of explicit logical facilities. In [Mellish, 1985], Mellish apologises for the need to maintain all these different logical entities denoting the same discourse world entity(ies), and, quite correctly, justifies it on account of the existence of the different candidate sets for different references before those references are fully resolved. But the only reason an apology is necessary is because of the confusion between representation of linguistic references and of discourse world entities.

Finally, it is Mellish's stated intention to treat indefinite entities and reference entities with the same mechanisms. While this seems a sensible aim, it is hard to understand, in that reference entities are associated with candidate sets and indefinite entities are not (rather, they (together) constitute candidate sets), and it is the constraining of the candidate set which captures the referential behaviour of a noun phrase.

If Mellish were to say that his indefinite entities represented introductory referring expressions and to give another level at which the actual discourse entities were to be represented, he could account cleanly for candidate sets in terms of the division into those same levels. The issue of representing the singular or plural surface form of reference to sets would also be diffused, because he would be representing referring expressions at one level, and referents at another, so syntactic markings could be safely made at the former level without polluting the latter. And indefinite, set and reference entities could all be treated in the same way, because they would all have (in some cases degenerate) candidate sets.

This is the approach taken in George.

4.4. Representation of Surface Structure and Reference

It seems, then, that the intention of both Mellish and Webber is that there should be a division between language and discourse world. What is not so obvious is whether either of these researchers made the correct choice in their representation – I argued in the last section that they did not, whether deliberately or otherwise, mainly because of their use of amalgamated representations. The view embodied in the George system and theory is also that there should be a division between representation of language and representation of discourse world, on abstract philosophical grounds; however, it is the stronger view that such a division should be as complete as possible. I have argued this point at various

stages in this document. The choice of a strongly separated representation implies four basic requirements, which are as follows.

1. There must be two distinct (though not necessarily different) languages for representing the discourse and the hearer's model of the world it describes.

2. There must be an explicit connection between the two representations – otherwise it would be impossible to represent the full meaning of a discourse. This meaning is not expressible within either of the distinct representations, because it includes elements of each, so must be external to both.

3. This connection can never be discarded, because it is never possible to take objects in the world and substitute them into the language.

4. There must be either an inference mechanism over the combined languages, or a means of translating the combination into an alternative self-contained conventional form, to confer a meaning upon the representation (which is always true of non-standard representations).

The approach of making this division in general carries with it at least the following five advantages.

1. On a theoretical level, it removes the problem of distinguishing between names of objects and the objects themselves: all the logical objects on the linguistic side are names (at the meta-level with respect to the discourse world, if one cares to see it that way), while all those on the discourse world side (even, in principle, relations between discourse entities) are objects in the representation of that discourse world.

2. It is possible to represent linguistic features which determine referential behaviour without affecting the nature of the discourse world; thus, for example, the distinction in behaviour between "Each" phrases and ordinary plural referring expressions can be represented in the linguistic representation without polluting the discourse world.

3. It is easy to represent candidate sets of referring expressions, and the representation has a clear status, distinct from both the discourse and the discourse world, in the connection between the two representations.

4. There is never any requirement to substitute entities for referring expressions, and so no decision about reference is ever irreversible. In the event that re-evaluation of reference becomes necessary, it is always possible, because no information is ever lost

through substitution. It is possible (and easy) to manipulate the mapping between the discourse and the discourse world without changing either of them.

5. It is easy to represent the obscure notional division between information which defines entities (such as "donkey-ness"), information which selects entities (as in "the donkey on the left") and information defining the situation described by a discourse, which is highly desirable from the point of view of analysing reference.

We can also argue that, if we are attempting to represent linguistic structure, we need to include in our representation aspects of language like subordination and definiteness, simply for completeness – otherwise it is harder to claim that what we have really is a representation of the discourse, and not some more intensional logical object. This is ideal for the representation of a discourse with a view to reference analysis.

Use of the George Representation Language, as described in Chapter 4, together with bindings to entity tokens, as described in Chapter 6, accords with the four basic requirements, and also the final suggestion, above, of a "complete" representation of discourse features and structure. This design decision is primarily motivated by a desire for a pure (*ie* non-amalgamated) representation of discourse into which the notion of candidate sets would neatly fit, via the mapping between discourse and discourse world.

However, there are more advantages still to such a "complete" representation when one works in a context of adaptable representations. In particular, when one is representing partial sentence semantics ambiguously, it will usually (one would hope!) be necessary to disambiguate the representation before the end of the sentence, or at least before the end of the discourse. Now, there are important cases where information about linguistic structure is largely irrelevant. One such case is the representation of unresolved definite noun phrase reference by candidate sets, as proposed and studied by [Mellish, 1981]. This is so because the semantics is represented as a series of constraints which are repeatedly augmented, in parallel with a repeated reduction in the size of the candidate set; this requires no change to the linguistic representation other than the addition of new sort information from the input, which is part of the standard parsing process anyway.

Consider, in contrast, George's treatment of noun phrase post-modifiers. In Chapter 5, I showed how a single representation can be used for the two categories np and np/nmod, given a suitable coercion to map from it to the correct category, which is determined by the subsequent word in the parse. Essentially, if the subsequent word does not contribute to a noun phrase post-modifier (nmod), the ambiguous representation (which is of category np/nmod) is coerced to an unambiguous one of category np.

The relevance of this to the argument here lies in the structure of the ambiguous representation. For reasons argued in Chapter 5, the ambiguous syntax:semantics pair representation for np and np/mod is this (where **Q** is a possibly empty string of quantifiers and ρ is a GRL reference):

$$\text{np/nmod} : \lambda v.\ \mathbf{Q}.\ [v,\rho] \blacktriangleright \rho \qquad 238$$

As it happens, this representation is the same as the unambiguous representation of a post-modified noun phrase. Now, we want to coerce this to a noun phrase by removing the information specific to post-modifiers from the semantics. This we simply could not do if those parts of the representation were not in some sense explicitly visible. Note also that we cannot dismiss this issue as arising from the particular grammar implementation in George, since, as I explained in Chapter 5, there is no other way to represent this ambiguity in a strictly incremental parse (*ie* without backtracking and/or the ability to re-decompose partial representations).

A perhaps more obvious benefit of exhaustive linguistic representation is the ability to distinguish between definite and indefinite noun phrases. George uses the (acceptable but generally inadequate) assumption that indefinite phrases introduce new entities (or rather entity tokens, in George terminology). In the event of referential failure, definites and indefinites need to be treated differently. George's parsing algorithm allows the candidate sets of definite references to become empty if the context so requires, and responds by creating a new entity token, on the grounds that the reference was really introductory; and, in particular, this can happen some time after the phrase was parsed. On the other hand, if the candidate set of an indefinite (*qua* introductory) reference becomes empty, an error is flagged, because the discourse state is therefore such that the phrase cannot refer to the entity which it introduced – which is necessarily wrong. Without explicit representation of definiteness, such behaviour would not be possible. I emphasise again that this is an adaptability requirement; the notion of introductory definites is not admitted in Mellish's early/incremental theory, and it is not clear that the idea of adding new entities to candidate sets fits comfortably within that view of the reference problem.

Note again, incidentally, that there is no question of making Hirst's confusion between antecedent entities and introductory referring expressions here. The two are clearly separated, one in each language.

In the next section, I will demonstrate the still greater adaptability required in order to reason about underspecified sets.

The main message of this subsection, then, is that for a representation to be truly adaptable, there are two fundamental and absolute requirements. First is a separation between discourse and discourse world, in order that either may be manipulated without disturbing the other. Second, but no less important, is the recording of a maximal amount of linguistic information, in order that it be possible to define preconditions and postconditions for whatever structural coercions are necessary in terms of the linguistic surface form representation. The four requirements listed at the beginning of this section arise from these.

4.5. Existential Quantification

In [Webber, 1979], Webber uses the familiar existential quantifier of (presumably classical?) logic to denote the semantics of indefinite (*qua* introductory) noun phrases. Thus, "A man runs." is translated thus, assuming the noun phrase is identified as non-generic and introductory:

$$S_1 \;\; \exists(x{:}\mathrm{Man}).[\mathrm{Runs}\ x]$$
$$e_1 \;\; \imath x : \mathrm{Man}\ x \;\&\; \mathrm{Runs}\ x \;\&\; \textbf{evoke}\ S_1, x \qquad 239$$

Now, as we saw in the example quoted in Section 4.4, when the change is made to a Level 2 representation (in Webber's framework) the entity name is not substituted for the introductory existential, on the grounds that the initial existentially quantified variable is the introduction of the entity. This kind of notation for introductory phrases, or, indeed, simply for indefinites, is very widespread in the linguistics community.

Let us consider the logical implications of this. First, are we making a strong claim about the meaning of introductory phrases? Does "A man..." really mean "There exists a man...", and is it false if there is not? If so, why do we need constructions like "There is a man...", where there is an explicit existential content?

Also, consider the effect of negation on existentials. If we negate an existential logical expression, we get "There does not exist an x..." or "For all x, it is not the case that..." which constitute the negation not of "An x..." but of "There is an x..."; again, the existential is rather stronger than in the plain indefinite. In "A man did not come in" there is even a reading where the entity said to exist is something other than a man (*eg* "I

opened the door, expecting the brush salesman. However, a man did not come in. The vendor was a big hairy monster.").

An alternative view of this use of the existential is that it is really just a syntactic declaration for a variable (or, more generally, logical object) which names an entity in the discourse world, and that it specifies the legal scope of that variable. This view, too, has its problems. It means that (at least in a purely logical representation) any subsequent reference to the same entity will need eventually to be represented by the existentially quantified variable which names it, unless the representation of reference is left to the assignment in the model representing the discourse world, which seems a counter-productive solution, because of the resulting complication in performing inference. Thus, the scope of the existential is not the sentence in which it appears, but the whole of the subsequent discourse. This has undesirable implications for those linguists who would like to generate different readings of sentences containing "each" by manipulating quantifier scope. Note that Webber's approach, above, is one stereotypical solution to this problem; but that it uses the extra-logical facilities of her representation.

Why, then, do linguists so frequently represent introductory phrases with existential quantifications? Clearly, one answer lies in history – because the existential quantifier was already defined in classical logic, we have attempted to force natural language semantics directly into such a framework, simply because nothing better existed. This attempt, it must be said, is not unmotivated – there are advantages to representing semantics in a well-understood system, not least the existence of a standard, well-understood inference system over (and/or semantics for) the language. There are of course other, potentially more task-specific benefits, such as (some linguists would say) the ability mentioned above to represent different readings of some utterances through different combinations of quantifier scope over the same expression.

Even so, I would suggest that the association of truth-functional semantics with a linguistic effect which is primarily pragmatic – namely, the creation or moving into focus of a discourse world entity – is misguided. Thus, the representations of such utterances as "there is a donkey in the jacuzzi..." are justifiably existential, but the representation of utterances such as "a donkey in a jacuzzi..." are simply references which happen to have the property of (usually) introducing discourse entities.

This view is particularly appropriate in George's context of linguistic representation, in that we are not, anyway, attempting to represent truth functional semantics, but something much closer to the discourse itself. Again, looking at Webber's and Mellish's work, we find an implicit treatment of the same issue. Webber introduces her discourse

entities, and substitutes their names for subsequent reference to the object of the existential; there is an explicit connection between the original quantified expression and the entity (via **evoke**), which seems to serve no purpose than to perpetuate the quantifier; therefore, since Webber certainly does not appeal to the quantifier's logical meaning, the two might as well be conflated. Mellish uses his indefinite entities – ignoring for now the issue of whether they are linguistic or discourse world entities, the fact remains that they are quantifier-free, and neatly capture the weak existential nature of indefinites simply by existing in the representation system.

This is the approach taken in George. Representation of "strong" existentials is left as further work.

Further, in Chapter 7, I argued that the use of the classical universal quantifier alone is inadequate to represent the multiplicity of plural and quantified references which exist in English, and that we really need at least two kinds of quantification for effective representation of natural language. Given this, the conventional use of quantifier scope to generate different readings cannot necessarily be depended upon, and is therefore less effective as a justification for existential quantification.

The final argument against the (straightforward) use of the existential quantifier for representing language semantics is the behaviour of the familiar "donkey sentence". I will discuss this in Section 5 of this chapter, since the effect is related to set reference.

4.6.Summary

In this section, I have made the following points.

1. (Section 4.2) If we wish to represent discourses and the situations they describe on computers effectively, we must be very clear about the theoretical distinction between the two.

2. (Section 4.3) Amalgamated representations, where representations of the discourse are mixed with those of the world, implicitly or explicity, deliberately or otherwise, can lead to over-complicated, difficult-to-manipulate or just plain wrong analyses.

3. (Sections 4.3 and 4.4) Given the aim of explicitly representing the discourse, and not just its semantics, it is quite easy to justify inclusion of linguistic information (*eg* definiteness) which might not normally be included in a semantic representation, but which can be useful.

4. (Section 4.4) Separated (*ie* non-amalgamated) representations are particularly suitable for use as adaptable representations, because the separation admits independent manipulation of the discourse and of the world.

5. (Section 4.4) Separated representations are suitable for use as adaptable representations, because they facilitate storage of a maximal amount of linguistic information, which is needed to guide coercion (see 3, above).

6. (Section 4.4) Both [Webber, 1979] and [Mellish, 1981] take steps towards separated representations, but seem to regard this, or its consequences, as failures in their systems. Webber hints at the desirability of adaptable (or at least incremental) representation; Mellish's work hinges on the idea of incremental evaluation, but does not extend to issues requiring full adaptability.

7. (Section 4.5) The use of the classical existential quantifier in conventional semantic representations to represent the introduction of an object (by binding a corresponding variable) is arguably misguided. A better approach is to use the existence of an object in the representation to denote introductory reference.

5. Representation of and Reasoning about Set Reference

5.1. Introduction

The issue of set reference in discourse is one which has received surprisingly little attention in the work of Natural Language researchers. Here, of the work singled out for attention in Chapter 2, we again need primarily to discuss the work of [Webber, 1979] and [Mellish, 1981] – for [Haddock, 1989]'s work relates only to singular noun phrases.

By way of further restriction on the domain of data under consideration, I have covered only sets introduced and referred to by what we might call quantified references. I will not cover those referred to or introduced by collective nouns, because the syntactic and semantic behaviour of such references is quite different from that of the quantified ones, and does not fit into the framework proposed here (note, though, that the framework can in principle be easily extended).

It should be remembered (from Chapter 1) that the primary goal of this research is to produce a computational account of reference to underspecified sets and their subsets; as

such, the work stands alone, since none of the related work reported here addresses the same issues; indeed, I am not aware of any existing work which does so.

5.2. Sets and Quantification

5.2.1. Introduction

The first issue to discuss here is that of quantification. I argued in Section 4.5 that the existential quantifier is not necessarily the best way to represent non-explicit existentials in language. Also in that section, I referred to the argument in Chapter 7 that there are at least two different kinds of "universal" quantification in natural language; for purposes of simplification I have considered here just two, characterised as strong and weak quantification, corresponding with "each"- and set-type quantification in English and with strong and weak indexing in the George Representation Language.

Both Webber and Mellish (as well as many others) acknowledge a distinction corresponding with this one. Webber, however, is unusual in that she attempts to represent it explicitly in her translation; Mellish represents it explicitly too, but arguably for some of the wrong reasons (see Section 4.3). In this section, I will explain these two views in detail. The main point is that quantification in George behaves as a function applying to references to produce, effectively, new references; the other approaches do not explicitly subscribe to such an exotic view.

5.2.2. Webber's Approach to Set Reference

[Webber, 1979] takes a fairly conventional means of representing sets based on universal quantification. To maintain uniformity, she needs a type system for sets like that already specified for individuals (which I introduced in Section 4.3). Such a system may be generated from the existing system by means of a parametric type constructor, which Webber calls **set**. Webber views **set** simply as a function mapping from a type with an extension E to a new type whose extension is the power set of E, 2^E. She also defines a related function, **maxset**, which performs a similar mapping, but such that the extension of **maxset**(T) is the largest member of 2^E, where E is the extension of T.

For example, in Webber's own terms (where L stands for "lifted"):

"if

λ(v:Man)[($\exists$y:Piano). **L** v, y]

is a predicate true if its argument is an individual who lifted a piano, then

λ(v:**set**(Man))[($\exists$y:Piano). **L** v, y]

is a predicate true if its argument is a set of men such that the set of them lifted a piano. On the other hand,

set(λ(v:Man)[($\exists$y:Piano). **L** v, y])

is a predicate which is true if is argument is a set of men such that **each** of them lifted a piano. "

Webber goes on to show how these predicates can be used as types of variables to restrict the range of existential and universal quantifiers, as for the base types (see Section 4.3). She also emphasises again differences in interpretation corresponding with differences in quantifier scope:

"Indefinite plurals can be represented just like indefinite singulars using the existential operator and an appropriate predicate for the quantifier restriction. For example,

(i) ($\exists$x:λ(v:**set**(Man))[($\exists$y:Piano) . **L** v, y])
"some men who (together) lifted a piano"

(ii) ($\exists$x:**set**(λ(v:Man)[($\exists$y:Piano) . **L** v, y]))
or
($\exists$x:λ(v:**set**(Man))[($\forall$u$\in$v)($\exists$y:Piano) . **L** u, y])
"some men who (each) lifted a piano"

Definite plurals can be represented like definite singulars using the definite operator and either the **set** or **maxset** function.

(iii) ιx:λ(v:**set**(Man))[($\exists$y:Piano) . **L** v, y]x
"the men who (together) lifted a piano"

(iv) ιx:**maxset**(λ(v:Man)[($\exists$y:Piano) . **L** v, y])x
"the men who (each) lifted a piano"

In (iv) the definiteness of the plural is captured by the fact that the maximal set of individuals satisfying any given predicate is always unique."

One criticism arising here is the production of two different readings for the same expression in (i) and the first reading of (ii). The two readings denote different interpretations: collective and distributive, respectively. Now, while Webber is certainly correct that these two are distinct readings with different logical properties, it is not clear that representing them as such as soon as they appear is a constructive thing to do. Here is one issue where adaptability can be a real bonus: the collective/distributive distinction arises in the majority of plural references, and, if represented explicitly, can easily lead to a combinatorial explosion of readings of a discourse. If we allow an adaptable, ambiguous representation of sets, which may be coerced to a more specific one when we have grounds for so doing, we can remove a large amount of potential non-determinism from our system. What is more, it is easy to construct examples where this particular distinction makes no difference at all to the meaning, because inferences based on it are not required.

Therefore, it seems doubly pointless to represent it explicitly, at least in the first instance. This is the approach, made possible by adaptability, taken in George. (In fact, the distributive/collective distinction has been largely ignored here, on the grounds that this postponement is possible, and that the index partition and expansion operations can give rise to explicitly distributive readings where this becomes appropriate. Beyond this, the position is that the indexed reference is an indeterminate representation for either reading.)

Webber does not explain her separate representation of collectives and distributives until she discusses universal quantification, later on. This time, she includes explicit number information, in a way justified because

> "it might be useful to indicate it so as to be ignorable when identifying candidate antecedents for "one"-anaphora".

Webber comments that the use of the **set** type instead of the universal quantifier may be regarded as counter-intuitive, but points out that English obviously does maintain the distinction (if one views the universal quantifier as denoting distribution of properties). In particular:

> "Adopting the above conventions permits a separation of the notions of focussing the listener on a set of things and of saying something about that set or about its individual members. To attribute some property to each member of some set, I would merely add in a universal quantifier cf
>
> 49. Three men ate a pizza.
> $(\exists x{:}\mathbf{set}(\text{Men}))(\exists y{:}\text{Pizza})$. Ate x, y & $|x| = 3$
>
> 50. Three men each ate a pizza.
> $(\exists x{:}\mathbf{set}(\text{Men}))(\forall w \in x)(\exists y{:}\text{Pizza})$. Ate x, y & $|x| = 3$
>
> 51. The three men ate a pizza.
> $(\exists y{:}\text{Pizza})$. Ate $\iota x{:}\mathbf{maxset}(\text{Man})x$ & $|x| = 3$, y
>
> 51. The three men each ate a pizza.
> $(\forall w \in \iota x{:}\mathbf{maxset}(\text{Man})x$ & $|x| = 3)(\exists y{:}\text{Pizza})$. Ate w, y"

These examples raise a number of issues. First, Webber uses **maxset** to denote definiteness, and to echo the requirement for singular definites that the candidate entity is somehow unique. She also requires that number information be represented separately from type information, so that the two can be considered separately. She considers plural references as references to sets, and sometimes explicitly as references to individual members of sets. All of these decisions seem justifiable in isolation, and are all present as assumptions in the George DeReferencer; and it seems to me that the **set** type models very neatly what seems to be going on, and is therefore not counter-intuitive at all. Maybe

Webber feels that the traditional reliance on logical quantifiers I mentioned earlier is somehow more aesthetic.

However, it is noteworthy that Webber makes no attempt to deal with the kind of examples of underspecified set reference discussed in Chapter 7. This is probably because her interest is focussed on "what the discourse makes available for reference" – and, while underspecified sets certainly fit in this category, it is arguable that candidate entities for the kind of set reference I am covering here are not made available directly by the discourse, but by some kind of inference from it (which was the subject of Chapter 7). Also, though it seems hard to contradict the design decisions made above, some of them do themselves beg questions, not addressed by Webber, which seem unavoidable.

For example, take the issue of number representation. Webber claims to be representing something close to the surface structure of a discourse, including, for example, subordinate structure in her compound types, and that this is a particularly desirable feature of her system. Here, however, she takes information imparted by a quantifier and gives it the same status as a main clause in the representation. One would hope that this does not arise from a strong claim that the translation of "The three men" is the same as that of "The men ... and there were three of them" (*cf* Webber's justification of the **evoke** predicate), and as such is not so unreasonable; however, it does bring us back to the apparently unavoidable issue of the division between the information defining and distinguishing objects in a discourse world and that specifying the situation described by a discourse. It seems unlikely that one would ever in general be able to draw a clear line between these two. However, I suggest again that the division Webber makes (explicitly, with importance attached) is at best sloppy and at worst incorrect.

Having said this, I suggest that one area in which Webber is very distinctly correct is in her attempt generally to represent the kind of a reference (*viz* simple plural or quantified reference) made to an entity in the logical form of her translation. This is particularly important in the use of the sets and universals to denote distinction between references to sets and their individual members respectively. What is more, the representation allows explicit composition of sets from subsets along certain dimensions defined by the discourse, which Mellish shows to be desirable (see Section 5.2.3), because it is possible to associate a ∀ with each quantifier in the input, and view the whole set as the cross product of the subsets over which these quantifiers range.

In summary, then, Webber's view of quantification is a fairly conventional, intuitive one (even if she feels otherwise!). Sets are viewed as individuals or sets of individuals, depending on the form of the reference; selection of individuals in the latter circumstance

is achieved through universal quantification. The approach to numbered sets is again the obvious one: numbers are included in the logical form if they are present; otherwise, they are not.

One important part of the domain of reference in English that [Webber, 1979] does not cover is that of reference to subsets of the sets denoted by her entities. In particular, no notion of the elaboration of the detail of the relationships between individual members of sets taking part in set predications is specified, or even acknowledged as necessary. This is particularly disappointing, as it is easy to see, for example, how interactions between juxtaposed "each" references emerge correctly from Webber's representation.

5.2.3. Mellish's Approach to Set Reference

As I explained in Section 4.3, Mellish views set references as a special case of references, and introduces special entities into his world model to represent them (I have already commented on the philosophical issues of amalgamation inherent in this). Each *set entity* is annotated with a flag indicating whether or not it was introduced by a singular surface form (in particular, a noun phrase quantified by "each"); the presence or otherwise of this flag determines the required surface form of subsequent reference entities denoting the same set (either plural, for standard set references, or singular for "each" references). So, in Mellish's model, there are set entities, which may appear in candidate sets for set references. The surface form flag determines how the reference is treated in terms of its interaction with other references, which I will explain below.

Each of Mellish's entities is associated with a *dependency list*. This is a collection of pieces of information specifying how (if) the entity relates to other entities in the world model, and specifically those related by explicit predications. The entity representing the referent of 240a (from [Mellish, 1985, pp64-66]) is shown in 240b.

"...two blocks each containing 3 pulleys..." 240a

set entity: set(1),
elements: {p1, p2, p3}
dep list: [(external,3,F),(block1,2,F)] 240b

The example noun phrase introduces six pulleys (three in each of two blocks). The reference entity standing for the pulleys is therefore dependent on that standing for the blocks – otherwise there would be only three pulleys. Now, Mellish states that

"In general, an indefinite NP entity can decompose into separate dimensions for

1. The possible "external" set elements.
2. Each distinct universal quantifier that governs it."

These possible decompositions are denoted by the dependency list. In this case, there is one division into two, arising from the set entity "block1", and a division into three arising "externally" from the information explicitly given in the introductory noun phrase. In general, the number in the dependencies may be unspecified, simply because it is not calculable from or present in the language input (*cf* in Webber's system: the modulus part of her set description is optional).

The third slot in the dependency list is a usage indicator, taking values T or F, denoting "true" or "false". Mellish tells us that

> "This is a flag that tells whether this dependency has been the basis of a non-trivial division into sub-classes."

This is a distinction between a noun phrase like "two particles of mass a and b" and simply "two particles"; in Mellish's terms the latter division is trivial, whereas the former is not, and is based on a particular dependency between the particles and their masses (which are also world entities in Mellish's model). Now, Mellish goes on to say that

> "A dependency cannot possibly be marked "used" unless its number has been established, for it is impossible to consider the subsets separately until it is known how many there are."

I suggest that this analysis is incorrect, on the following grounds. While it is certainly the case that we cannot identify all the elements of such a divided set if we do not know how many there are, we can certainly pick out some subsets or individuals to leave a set whose number is still unknown. For example:

> "Some blocks, each containing some pulleys, are placed in a line on a table.
> A light cord is threaded into the leftmost pulley of each block." 241

Here, there is a non-trivial subdivision, in which one pulley acquires a unique distinguishing property, which is not dependent on how many pulleys there are. The pulleys are therefore non-trivially dependent on the blocks, and the cords are non-trivially dependent on both, even though no number information is given in either dependency. The reason that this example works is that the "leftmost" property always selects exactly one entity in a collection.

This point leads us on to the issue of notating choices between subsets and individuals within a set. Mellish's solution to this is

"the syntactic device of *subscripting*. This has the advantage that

- it can always be determined syntactically what relation an entity has to one of its sub-entities
- sub-entities can be used only as needed, without an explicit step of 'creation'."

The idea, embodied in the dependency list, is that sets of entities can be split up into subsets along *dimensions*; that is to say, along some measure indexed by the quantifier(s) applying to them. Thus, if we have three pulleys, say, we can refer to them as pulley number 1, pulley number 2, and pulley number 3; then, if there are two wheels in each pulley, we can label those as number 1 and number 2, but we must do so with respect to context – *ie* which pulley they are in. Thus, in Mellish's terms, wheel number is one dimension, pulley number is another. This idea corresponds with George's indexing.

What Mellish's arrangement does not give us is the ability to divide the range of an individual subscript, so that it is possible (*eg*) to refer to subsets of a particular uniform set, whose members may be indistinguishable, as in "Some of the men were tall... The others were short...". In order to do this, Mellish would have needed to introduce arithmetic relations between his subscripts, and to allow more than one subscript to range over each dimension, thus taking a step towards true adaptability of set representation – as it is, one can only represent incrementally what becomes explicitly available as the discourse proceeds, and not what is inferred from it. This more adaptable idea is the approach (with indices, index application and bounding constraints) taken in George.

Given this mechanism, in example 241 above, we could have a collection of unknown number and refer to "the leftmost" or some other such uniquely defining property by means of an arbitrary mapping from spatial arrangement to subscript (Mellish's term).

This very explicit approach to the "unfolding" of a set along a particular dimension leads to another question: that of the relationship between references joined by relations in a discourse, which I discussed in Chapter 7. Mellish takes a severely simplified view of this issue, the simplification being justified by the form of references in his domain of data (A-level mechanics questions). In general, however, Mellish's simplification is much too extreme.

The simplifying assumption is that relations between sets referred to by ordinary plurals are one-to-one correspondences, and that relations between sets referred to by "each" phrases are cross-products. Combination of the two kinds of reference results in a relation between each individual in one set and the entirety of the other set, whether distributively or collectively. The one-to-one correspondence case is denoted by the

existence of *linked dependencies* – dependencies associated with the two references whose subscripts have been unified. Thus, moving a subscript along a linked dimension in one reference results in a corresponding movement in the other. I will discuss this issue further in Section 5.

In Chapter 7, I presented a solution to the more general problem where relations between plural-referenced sets are arbitrary, which allowed us automatically to generate representations of the most specific readings possible, given a context, of such references. The approach allowed us to do so practically and efficiently, by viewing the representations as (initially) heavily ambiguous and (subsequently) adaptable, in order that the resulting potential combinatorial explosion should not present a problem.

Finally, adaptability was necessary in Chapter 7's analysis, where I allowed references to sets to be replaced first by references to unions of subsets, and then by conjunctions of references to those same subsets, in order to generate the underspecified mappings between plural references. This leads me to question Mellish's comment, reproduced above, that it was a good thing for sub-entities (representing subsets) to be used only as needed, without an explicit creation step. My objection is this. Webber has argued convincingly that we need to view reference analysis as the complementary tasks of finding what is available for reference, and then what refers to it. This comment of Mellish seems to contradict Webber's view completely, although he subscribes to it elsewhere. In particular, if we have referred to a subset of an existing set in a discourse, then that subset is much more firmly in focus (*ie* available for reference) than another to which we have not explicitly referred; in a working system, we certainly want to represent this fact in some way. It would seem impossible to do so without using a representation which is logically equivalent to creating a new entity to represent the subset. This last is the approach taken in George.

Alternatively, one might read Mellish's comment as saying that one does not wish to create entities before they are referred to – that is, a set should be thought of as an individual object unless or until one or some of its elements are referred to. This interpretation is effectively an argument for adaptability of set representation and is therefore completely in accord with the theory presented here.

5.2.4. The George Approach to Set Reference

In the foregoing chapters, I presented an alternative view of quantification, where predicates were applied to the members of sets by applying numerical labels, generated from $\mathbb{N}$, to referring expressions, rather than directly to entities in the discourse. Because there are (at least) two different kinds of quantification, there need to be (at least) two different kinds of application of the labels. In George, quantification is expressed through conventional universal quantification ($\forall$) over the natural numbers, and associated with the references themselves through the two index application operators, $\times$ and $\otimes$; the use of two operators is a simplification for research purposes.

Quantification and index application in GRL are equivalent to Mellish's subscripting in most ways (in terms of both what they do and how they do it). They result in exactly the same device of numbering individuals (although the numbering is achieved in George through manipulation of the surface form, rather than of the entities) and they allow the same division into dimensions, because each index corresponds with a quantifier, just as in MECHO each dependency corresponds with a quantifier. The justification for performing the numbering via the surface form (or George derivations from it) is simple: we are trying to deduce what is being referred to from the discourse, and to make the reference fit with what we already know (exactly as in Webber's characterisation of discourse reference). A neat way of doing this, given the basic ability to infer the existence of individuals and undivided sets, which we have in George, is to manipulate the discourse representation so that we are able to use that existing basic mechanism on the resulting (new) representation. This manipulation is the function of the operations on indices explained in Chapter 7. The manipulation might be viewed corresponding with the attempting to paraphrase ambiguous English into more specific versions.

To summarise, then, GRL and the associated manipulations are adequate for notation of [Mellish, 1985]'s approach to set reference (which subsumes [Webber, 1979]'s), but it also allows more. The index and entity token partition operations, coupled with the ability to rewrite expressions in the discourse memory, and with the entity token partition tree data structure (covered in Chapters 6 and 7) allow us adaptably to represent sets and reference to them in a motivated and well-defined way. Index dependency, which fulfils the same function as Mellish's linked dependencies, and the various index propagation operations, with the index application rule, give us the means of generating all the possible readings which Mellish rules out with his simplifying assumption about plural reference.

I must emphasise again that this approach to representation of set reference is entirely dependent on adaptability, in two ways. First, and most obviously, we need to be able to notate the fact that a relation which at first appeared to be quantified across the entirety of two sets may subsequently turn out to be much more complicated – the representation must therefore change, as it were, non-incrementally – information has not just been added; it has changed fundamentally. Secondly, if we were to allow explicit representation of all possible readings, the resulting explosion of non-determinism would probably be, in all but the most trivial cases, computationally intractable. The adaptability of this system allows us to represent all and only the information we have, in one (ambiguous) representation. Even so, this ambiguity does not detract from the correctness of the translation in the same way as ambiguity in resolving a definite reference; it is merely an acceptable vagueness. Thus, we might argue that ambiguity of the adaptable representation is justifiable, and even desirable, because it denotes underspecified references which are meaningful in spite of their underspecification. This seems to accord (in a loose sense) with the human ability to reason about underspecified sets. [VanLehn, 1978]'s conclusion regarding the postponement of scope evaluation adds strength to this argument.

5.3. The "Donkey Existential"

5.3.1 Approaches Based on Conventional Logic

In Section 4.5, I suggested that existential quantification is not the best way to represent introduction of entities by indefinite referring expressions. One of the strongest pieces of evidence for this view is the so-called "Donkey Sentence", 242, and the problematic indefinite referring expression therein, which I have called the "Donkey Existential".

"Every man who owns a donkey beats it." 242

A major part of the problem with the donkey sentence, quite apart from the analysis of its meaning, is in correctly representing that meaning once it has been deduced. One might at first sight expect 242 to be translated (in FOPC with set predicates; upper case variables range over sets) as something like 243.

* $\forall m.[\ man(\ m\ ) \Rightarrow \exists d.[\ donkey(\ d\ ) \wedge owns(\ d, m\ )\ ] \Rightarrow beats(\ d, m\ )\ ]$ 243

However, there is a scoping error in this translation – the last occurrence of d lies outside the scope of the quantifier which introduced it. The error arises from the attempt to

represent the dependence of the donkey's being beaten on its ownership by the man, and therefore on the donkey's existence (paraphrasing the original sentence as "If a man owns a donkey, then he beats the donkey he owns"). This is the essential problem with the Donkey Sentence; it must be tackled if we are to emulate the human reference analysis mechanism, which clearly has no difficulty with sentences of this kind. Three other possible logical representations are shown in 244.

* $\forall m.[\ man(\ m\ ) \Rightarrow \forall d.[\ donkey(\ d\ ) \wedge owns(\ d, m\ ) \Rightarrow beats(\ d, m\ )\ ]\ ]$ 244a

* $\forall m.[\ man(\ m\ ) \wedge owns(\ \delta(m), m\ ) \wedge donkey(\ \delta(m)\ ) \Rightarrow beats(\ \delta(m), m\ )\ ]$ 244b

* $\forall m.\exists o.[\ man(\ m\ ) \wedge \exists d.[\ owns(\ d, m\ ) \wedge donkey(d) \wedge o \equiv d\ ] \Rightarrow beats(\ o, m\ )\ ]$ 244c

In translation a, the existential quantifier, $\exists$, has been replaced by the universal, $\forall$, which ranges over the whole set of entities in the discourse (or at least those in focus, if we acknowledge that concept). The force of the existential is then maintained by selecting, as it were, each man and testing the ownership by him and the donkey-ness of each of the entities in the discourse. However, in the event that there is a donkey which a man does not own, this formula implies that he beats that donkey as well, by *ex falsio quod libet.* This does not capture the meaning of the English sentence.

In translation b, the function δ: men $\rightarrow$ donkeys is a skolem function; the expression is said to be *skolemised.* δ maps the each man on to the correct donkey. However, this just postpones the issue: in a real implementation, we may often need to know which donkey, and δ does not give us a way of calculating that. What is more, δ is a function, and so must be defined for all men – but the meaning of the sentence is such that not every man necessarily owns a donkey.We need something like $\exists\delta(m)$ in the condition of the $\Rightarrow$, and it is not at all clear what is the logical status and nature of such an existential.

Translation c uses a dummy variable to denote each donkey. However, if there is some object, not a donkey, which the man owns, the formula implies that he beats it, again by *ex falsio quod libet.* This does not capture the meaning of the English sentence.

All of these attempts at representation have in common the same problem: they do not capture the explicit conditional nature of the relative post-modifier in the original sentence. In particular, truth functional correctness is only achieved by default, as it were, through *ex falsio quod libet* on the implication. Thus, we can get away with the lack of clear meaning of the variables d and o in 244a and 244c and the inaccuracy of the function δ in 244b, because it no longer matters what is the truth value of the conclusion of the implication – the implication is vacuously true anyway. This is essentially unsatisfying,

because it fundamentally fails to capture the dependence of the beating upon the existence of the donkey as well as its ownership by the man.

One step towards a solution to this problem is to use a system which discriminates between explicit discursive use of the existential as in 245:

"If there is a spare carrot, the donkey will eat it." 245

(which is a simpler, more obvious form of the usage found in the donkey sentence), and the more conventional logical usage, as in 246, where, as I said before, it simply serves the same function as a variable declaration in a programming language.

"Jim is a man." 246a

$\exists j.[\, \mathit{called}(j,\text{"Jim"}) \wedge \mathit{man}(j)\,]$ 246b

This discrimination is in part the approach taken in George. In GRL, references are simply references; they are not variables denoting discourse world objects. They refer to (or rather, are bound to) entity tokens. If no binding can be made, and it is not appropriate to assume that the reference is introductory, then the discourse analysis has failed, in a procedural sense – truth and falsehood do not arise; in particular, there is no question of real existential statements representing introductory references. (Neither is there, however, a notion of explicit existential in GRL, meaning something like "existence in the discourse world" as defined here; this is a feature which, though clearly necessary for full generality, is not vital to the use of the system for studying underspecified set reference.)

5.3.2 Approaches Based on Multi-Level Representation

Webber's approach to the more specific question of the "donkey existential" is to see the donkey as a prototype. In a sense, the entity arising from that reference is the intension of the donkey, which extends neatly to the original, more complicated donkey sentence (242). Here, the prototype can be viewed as "the prototypical donkey which each man owns". Webber's translation of this is shown in 247; **M** stands for Man, **D** for Donkey, and $S_{73.1}$ refers to the subordinate clause.

"Every man who owns a donkey beats it."

Level 1: S_{73}: $(\forall x{:}\lambda(u{:}\mathbf{M})[(\exists y{:}\mathbf{D}).\text{Own } u,y]).\text{Beat } x,\text{IT}$

Level 2: $(\forall x{:}\lambda(u{:}\mathbf{M})[(\exists y{:}\mathbf{D}).\text{Own } u,y]).\text{Beat } x,\ \imath y{:}\ \mathbf{D}y\ \&\ \text{Own } x,y\ \&\ \mathbf{evoke}\ S_{73.1},\ y$ 247

Mellish takes a similar view, in that, given the inference that the quantification in the sentence applies to "a donkey" and "it" as well as to "Every man", MECHO-style set reference entities with linked dependencies can be introduced to represent the sets.

In George, the view is rather more uniform. The entity token bound to each reference, as always, specifies a set (whose number is, in this case, unknown). Thus, it can equally be viewed as the intension of the set of donkeys, and so serve effectively as a prototype. Then, we can think of the successive index operations applied to the reference as leading to "less prototypical" representations, all without changing our view of indexed references as maximally ambiguous representations, as the special kind of ambiguity in the donkey existential is resolved.

The conditional nature of the relative post-modification in the donkey sentence is captured in the notion of context extension, where the validity of the sentence in the discourse (and hence of the discourse itself) is dependent not only on the truth of the condition given suitable bindings, but also on the existence of the bindings themselves. Thus, the force of the existential is lifted out of the linguistic representation, which seems correct, because judgement on correctness of reference is external to the language of the discourse itself.

Webber's approach to this kind of reference (which she calls *dependent reference*) extends beyond just donkey sentences. Regrettably, though, while she gives a means of representing the forms, after they have been detected, she does not show how such detection is achieved. In George, adaptability (and at some points Crain and Steedman's Principle of Parsimony – see Chapter 2, Section 3) mean that explicit detection is unnecessary. For example, the singular "a donkey" reference of the donkey sentence is adapted to a set reference by strong index propagation; this in turn enables the adaptation of the "it" reference, and so on (see Chapter 7 for details). Thus, the prototypical nature of the reference is in a sense emergent from the behaviour of the system, and does not need to be spotted in advance; and, as I said above, the subsequent behaviour of a prototypical (or intensional) reference is anyway subsumed in George's standard adaptable behaviour.

5.3.3 An Approach Based on Discourse Repesentation Theory

In Chapter 7, Section 18.4, I explained how George's approach of rewriting expressions to explicitly represent the effects of quantifiers on each other gives an answer to the questions raised by [Kamp, 1981]. I have not, however, commented otherwise on the

relation between Kamp's work and that presented here. Part of the reason for this is that Kamp does not address the issues of reference to subsets of sets and underspecified reference. Nevertheless, it is worth giving a brief comparsion here, for completeness.

Discourse Representation Theory is a procedural attempt to explain inter-sentential reference in a framework also able to cope with intra-sentential reference. It focuses on problems of quantifier scope, as does the work presented here, particularly on difficult examples like donkey sentences. The major difference between DRT systems and George is that the output given by the DRT creation procedure is viewed as a logical form (though it is usually represented graphically), and that (English) quantifier scope is allowed directly to determine the form of an analysis. In particular, universal quantification (with its scope), and subordination both give rise to structures "correspond[ing] roughly to implication in first order logic" [Johnson & Klein, 1986]. I have already argued that conventional implication is not adequate properly to represent the implication-like connection between the sets of men and donkeys in the donkey sentence. An example, the DR state representing the donkey sentence, is given in Figure 19.

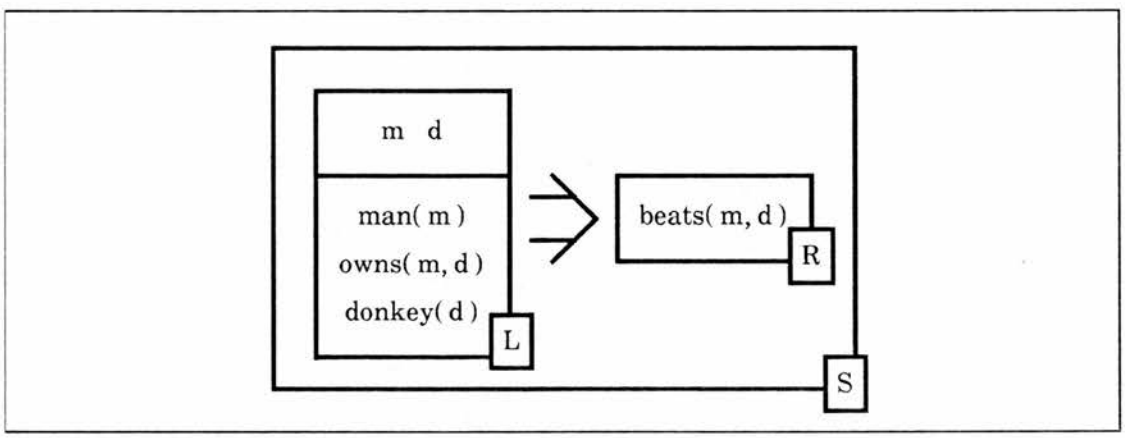


Figure 19: *DRT representation of "Every man who owns a donkey beats it."*

The point of using the boxes is to represent the scopes of variables, and their availablity for reference. In particular, a variable is available for reference in any boxes within the box in which it is declared (m and d are declared in box L in the example), and in any box which is a logical consequence of it – like R in the example, and nowhere else. Thus, m and d and available in boxes L and R but not elsewhere in S. This expresses the fact that the set of men, say, can only be accessed by a reference in the scope of the quantifier.

Note, however, that this tells us nothing of the actual relation between the men and the donkeys, and if we use conventional truth functional readings of the ⇒ operator, we have the same problems with derivation from false as before.

Further, this representation does not directly address the root of the peculiar referential behaviour of the dependent references. The issue arises not just because of the dependence (which is represented clearly by the DRT notation), but also because of the linguistic fact that a singular reference is being used to refer to a set entity. The correct intuition is that the plural nature of the quantification is being passed to the syntactically singular donkey existential, and therefore it is not subsequently available for binding to singular references. This intuition is expressed explicitly in George by the strong index propagation operation.

5.4. Representing English Quantifier Scope in GRL

In Section 5.3, I suggested that we should remove the existential quantifier from our language. What implications does this have with respect to what is left?

The point which most readily springs to mind is the representation of the scope of the quantifiers in the natural language input. As it turns out in GRL, we notate all quantification in our discourse representation with the application of labels generated by (possibly bounded) ranging over the natural numbers. How, then do we reproduce the effects of conventional quantifier scoping?

The simple answer to this is that quantifiers themselves in GRL co-relate in the most naïve and least interesting way possible. We have explicit mechanisms (index propagation) for expressing the interrelation of set references. Thus, subject to these mechanisms, the interaction of the (natural number) quantifications themselves is immaterial. This is expressed in the quantifier expansion operation defined in Chapter 4 and used in examples throughout this document to construct GRL expressions – no matter where quantifiers originate in an expression, their scope is the whole of that expression.

This obviously begs the question: why have quantifiers at all? Why not just assume the quantification from the existence of the indices. The easiest answer is that we wish to express bounds on the quantification, and we need a standard place at which to do so. Thus, we have a representation in GRL which has the necessary capabilities (like both Webber's and Mellish's) and which makes quantification explicit, in the form of indices, and therefore – and this is the crucial point – directly available for automatic

manipulation. Finally, by allowing the specification of upper bounds on the quantification (again like both Webber and Mellish), we have a device for dividing ranges of quantifiers (and in this George stands alone), which, in conjunction with arithmetic constraints on the bounds and rules for reasoning about them, allows us to express the kinds of division in sets which concern us here.

Finally on this issue, I must cover the issue of adaptability *per se* in quantified reference. The claim is that, to enable an adaptable representation of set reference, we need a representation in which the scope of English quantifiers is indeterminate. The simple reason for this is that many of the possible readings we wish to conflate into one ambiguous translation will be distinguishable (in conventional terms) only by quantifier scope. One way of doing so, is by the freely-generating algorithmic method of [Hobbs & Shieber, 1987], maybe augmented in the style of [Lewin, 1990]. These approaches have much in common with that in George, except that they produce logical forms from an initial intermediate form, rather than a more specific discourse representation from a less specific one. However, they do not acknowledge the idea of adaptable ambiguous representation, except in their initial supposition of a maximally ambiguous form from which all other legal non-ambiguous forms can be generated.

5.5. Underspecified Relations between and Reference to Sets

Finally, I must discuss the central issue of this thesis: the automatic generation of possible readings of sentences describing underspecified relations between sets. In Chapter 7, I showed how the George system enables complex derivations involving series of inferences regarding the nature of such relations, using the adaptability of the representation system to avoid combinatorial explosion.

The issue of reasoning about underspecified set reference has not been addressed before. In particular, both [Webber, 1979]'s and [Mellish ,1985]'s work stop short of the problem. It is not clear that Mellish's system is capable of this kind of reasoning without additions which would render it equivalent to George. None of this kind of reasoning is demonstrated in Webber's exposition, and, though it is fairly clear how the representation of these complicated relations would proceed at a general level, the details are obscure. I suspect that new mechanisms (*eg* set membership and subsumption) would be needed.

6. Summary – The Contribution of this Thesis

In this document, and specifically in this chapter, I have argued the following points (some of which are repeated from the earlier section summaries).

1. (Section 2.2) Use of *adaptable* representations can increase determinism in parsing systems.

2. (Section 2.2) Strictly incremental parsing removes the problem of so-called spurious ambiguity inherent in categorial parsing.

3. (Section 2.3) Use of adaptable representations need not rule out assessment of preference between readings, as required by some theories of language analysis.

4. (Section 2.4) To take advantage of implicit ambiguity in adaptable representations, a disambiguation mechanism is necessary. *Coercion* is one such mechanism.

5. (Section 2.4) The mechanism of Steedman's enhanced categorial grammars may be simplified by the addition of the *protraction* operation.

6. (Section 3.2) Early, incremental reference evaluation requires an adaptable representation.

7. (Section 3.2) Representation of and reasoning about predications between individuals within underspecified sets does not fit easily into existing incremental frameworks.

8. (Section 3.3) There can be no safe limit to adaptability, within a given discourse. The need for adaptability can easily extend beyond the end of a referring expression.

9. (Section 3.4) Adaptability of representation need not compromise our ability to analyse problem references of the kind explored by [Haddock, 1989].

10. (Section 4.2) In representing discourses and situations, we must be clear about the theoretical distinction between the two.

11. (Section 4.3) Amalgamated representations can lead to incorrect analyses.

12. (Sections 4.3 and 4.4) In explicitly representing the discourse, and not just its semantics, we can include useful linguistic information which might not normally be included in a purely logical semantic representation. This can be used both for

semantic and referential purposes and to aid parsing. Thus, in particular, separated representations are suited to adaptability, because they facilitate storage of a maximal amount of linguistic information, which is needed to guide coercion.

13. (Section 4.4) Separated representations are suited to adaptability, because separation admits independent manipulation of the discourse and of the discourse world.

14. (Section 4.4) Both [Webber, 1979] and [Mellish, 1981] take steps towards separated representations. Webber hints at the desirability of adaptable representation; Mellish's work hinges on the idea of incremental evaluation.

15. (Section 4.5) Use of the classical existential quantifier to represent the introduction of an object is misguided. It is better to use the existence of an object in the representation to denote introductory reference, with an explicit strong existential where necessary.

16. (Section 5.2) Purely logical languages, with only the conventional quantifiers, are inadequate to represent natural language discourse for the purposes of reference analysis. Both [Webber, 1979] and [Mellish, 1981] subscribe to this view. Neither Mellish nor Webber proposes a system which can analyse underspecified set reference.

17. (Section 5.2.4) The George system uses a notion (*indexing*) of application of quantification similar to that of Mellish's subscripting, but more general.

18. (Section 5.2.4) The correct behaviour of the George quantification mechanism is entirely dependent on adaptability.

19. (Section 5.3) Neither use of adaptable representations nor removal of the classical existential quantifier precludes correct analysis of dependent reference (*eg* the "donkey existential").

20. (Section 5.4) George's indexing mechanism allows adaptable ambiguous representation of readings conventionally differentiated by quantifier scope.

21. (Section 5.4) The use of adaptable ambiguous representations allows us to represent multiplicities of readings which would probably often be intractable otherwise.

22. (Section 5.5) The issue of underspecified reference to sets has been an open question.

I suggest that the work on set reference presented in this document is necessary (because underspecified set reference occurs frequently in real discourse), novel (because the

problems covered seem to be previously unaddressed) and at least partially correct (on account of the examples and justification presented in Chapters 6 and 7). As such, it contributes a first step towards a more complete coverage of natural linguistic phenomena than before.

The wider context of parsing and translating with adaptable representations presented in Chapters 3 to 7 also seems to contribute a useful new view. The idea is an extension or generalisation of the incremental approach of [Mellish, 1985], which has been justified here and elsewhere as psycholinguistically useful in its own right. Further, it is beneficial on a practical, computational level, in that it enables more efficient and less explosive parsing of heavily ambiguous utterances. On this basis, it provides a framework in which the set reference analysis mechanism can realistically operate; without ambiguous representation, the indeterminacy inherent in the mechanism and the data it covers would quickly get out of control; without adaptable representation – not necessarily the particular representation used here – such ambiguous representation would not be possible. Finally, it seems possible (and this is an area for further investigation) that this view could emulate the human ability to reason about underspecified sets.

There exists an implementation which embodies many of the ideas presented in this document. Its behaviour, at a simple experimental level, confirms many of the suppositions made in the foregoing chapters.

7. Afterword

This chapter has presented the contribution to knowledge of the George system and its associated theory. Even so, there are many more issues which might be studied in this context, some of which touch on ideas discussed in this document. In the next (final) chapter, I will briefly discuss possible future extensions and improvements of the theory and the working system.

Chapter 9
Further Work

Abstract

Various shortcomings of the Geogre system are discussed, and possibilities for further work based on it are proposed.

1. Introduction

This chapter is divided into two sections. The first discusses some theoretical shortcomings of the George system as presented here. The second follows up a number of issues which were left more positively in the main body of this document as subjects for further work. As such, it is merely a sketch of some interesting ideas, and should not be read as anything more. Most of the issues covered here are to do with extensions of the George system and theory to cover more linguistic phenomena.

2. Shortcomings of the George System

2.1. Introduction

In this document, I have presented a number of examples of manipulation of GRL expressions involving index propagation operations. In particular, some of these propagations were on to one occurrence of a reference appearing a number of times in the discourse memory. In these cases, it was necessary explicitly to ensure that appropriate indices and, if necessary, quantifiers, were associated with these other occurrences of the reference. Now, even though the success of the examples might be said to suggest that this is necessary, the existing characterisation is not a clean one. Consideration of this issue will raise a few related points, eventually leading (in principle) to a more general view of reference to quantified sets.

2.2. Representation of Universal Quantification

In Chapter 8, I cast doubt on the conventional use of the existential quantifier in representations of discourse, and on the traditional use of logical quantifier scope to represent interaction between references and quantifiers in language. In particular, I suggested that the scope of a universal quantifier was, in some sense, the whole of the expression in which it appeared, and that interactions usually expressed in terms of scope might be be expressed by applying and then manipulating GRL indices.

In fact, this approach is insufficiently general. The reality of the situation is that the scope of a universal quantifier needs to be not just the expression in which it appears, but (potentially) the entirety of the remaining discourse; this is the approach taken in the GRL-FOPC translation algorithm given in Chapter 4. To see why this is so, consider generalising the usual donkey sentence to the pair of sentences in 248.

"Every man owns a donkey. He beats it." 248

Here, the quantifying effect of "Every" has (in one reading) propagated not only to "a donkey" but also to the first noun phrase in the second sentence, "He", and thence to the second, "it". Thus, the notion of scope manipulation within sentences to represent readings is plainly inadequate for the task for which it is so frequently used.

2.3. Universal Quantification in George

As far as George is concerned, this realisation is not as painful as it would be for a system which used quantifier scopes to generate readings. It does, however, beg the question of how correctly to characterise referential interaction of the inter-sentential kind exemplified in 248. Already in George, there is the requirement mentioned above that propagation of indices be consistent for any one reference, and there is Rule 19 (Index Propagation through Context Extension), both of which have some flavour of the kind of operation we need here. In the next section, I will suggest a rule which will subsume example 248 and the more usual donkey sentence.

First, though, we must address the issue of quantification itself. In this document, I have presented quantification as something local to a given sentence, and I have used a fairly conventional quantifier notation (∀) to do so. The question now is whether such a localised notation is appropriate at all.

In general, the philosophy in George has been to maintain a representation close to the surface form of the linguistic input, or at least something derived directly from it by index manipulation. The justification for this is that in an adaptable system we wish to keep as much information as possible about our input, to facilitate adapability. On this basis, then, one might make a good argument for keeping an explicit quantifier notation on the GRL form of the sentence which introduced the quantifier, to record where the quantification came from.

The quantification information itself, however, needs to be more generally available, and there already exists in George an obvious place to store it. The information expressed by the existing indexed form is: what kind of quantification is applied (strong or weak), which references it is applied to, and what (if any) upper bound it has. The first of these is of necessity determined in the reference itself. The second is represented by the index symbol which is unique for each quantifier. The third is represented by the upper bound associated with the quantifier.

In the George system described here, there is a database of arithmetic equations between these upper bounds – the Bounding Constraint database. There is an operator, *upb*, which, when applied to an index symbol, returns its upper bound. One obvious way of representing the quantifier is to place an equation between the *upb* of each index and its value in that database as a bounding constraint. This then places solution of the equations resulting from index manipulations firmly into the domain of numerical constraint satisfaction, soluble in well-understood ways such as the Simplex algorithm. The kind of complex manipulations performed on indices and their upper bounds in Chapter 7 immediately become rather more accessible.

2.4. Inter-Sentential Index Propagation

Finally, then, we need a rule which will constrain propagation of indices between sentences to generate the correct readings.

First, the rule applies only to strong indices – ordinary (weak) plurals simply do not exhibit this effect: consider "Some men own a donkey. He beats it." where the effect of 248 does not occur. Second, we need a notion of topic or thread – consider "Every man owns a donkey. The women like him." where the interaction between "Every man" and "him" does not occur. Thirdly, as shown by the last example, there must be an intersection

between the sets of entities with sorts compatible with the two references (regardless of apparent number considerations).

All this sounds remarkably like Rule 19, except that, instead of requiring an explicit link by context extension, the only constraint is that the reference to which the index propagates should have appeared at the input after the quantifier. It would also seem that the within-sentence strong index behaviour already described would apply here too, since, in 248, the "it" reference may or may not be affected by the quantifier, as in my treatment of the donkey sentence in Chapter 7.

The last factor involved would presumably be a preference for this operation which decreased with time – it seems unlikely that a quantifier would normally affect a reference in another sentence when several unaffected sentences were interposed.

2.5. Some other Shortcomings

There are many factors in George which could be improved. I will suggest approaches to some of these in Section 3.

The general linguistic coverage of the coercive parser needs to be extended. One particularly interesting class of examples which would seem to present a problem for the coercive approach are those involving coordination. Also, surface reference anaphors (*eg* "the former", "the first", and reflexives) need to be covered – I will suggest an approach to reflexives later.

Another large hole in the explanation is the lack of discussion of the level of Discourse Entities, where the intensional reasoning about entity tokens is mapped into the discourse world. Again, I will discuss this briefly in the next section.

Finally, the whole issue of parsing with coercions and adaptable representations is worthy of considerable further investigation. In this document, I have listed a number of linguistic cases which are candidates for ambiguous encoding (*eg* various meanings of "is", and verbs with variable numbers of argument slots). Linguistic phenomena involving crossed composition in other CG frameworks should also be looked at further, and the backward version of my protraction operation (introduced in Chapter 5) should be investigated. In general, any linguistic behaviour involving stacking of constituents in conventional shift reduce parsers must be investigated, since embedment of such constituents within self-contained partial translations, and their resultant unavailability, in an incremental parser is always a potential problem.

3. Extensions to GRL

3.1. Introduction

The rest of this chapter is rather more positive. It proposes various extensions to George, some of which cover shortcomings listed above. In particular, in Chapter 4, I defined a representation language, GRL, which, in conjunction with a multi-level view of discourse representation, allowed implicit representation of ambiguity in discourse. In Chapters 6 and 7, I showed how the features represented explicitly in that language (sort, definiteness, strong and weak index application) allowed symbolic manipulation under various rules to generate possible readings of heavily ambiguous set reference.

In defining the representation language, I deliberately ignored some features which must be present in a general language understanding system, because they were not relevant to my presentation, and would merely have clouded the issue. In particular, the following important classes of utterance were excluded from George's coverage: explicit existentials, interrogatives, negatives, imperatives and compound utterances (by which I mean utterances containing conjunctions, disjunctions, implications, and so on); there are of course many others.

The main basis for the design of GRL was that the linguistic features used in determining reference should be explicitly represented (hence, *eg*, the definite operator, !). This approach seems to have proved fruitful. Therefore, the approach to covering the above issues in GRL would be along the same lines.

The final point regarding extensions to GRL here is the improvement of the GRL notions of abstraction. In particular, I will suggest a means of improving the language so that some λ-abstractions are replaced by an extended and improved notion of # abstraction, including unification. This will admit application of selectional restrictions on verbs.

In this section, when I refer to GRL, I mean the language as presented in Chapter 4. Extensions will be named GRL_X where X is a comma-separated string of symbols standing for the extensions (*eg* $GRL_\exists$ is GRL with explicit existentials).

3.2. Explicit Existentials

At various points in this document I have argued against the use of the conventional logical existential quantifier for the introduction of variables into expressions. This is because of the difference between (eg) "A man..." (only weakly an existential statement) and "There exists a man..." (a strongly existential statement). For this reason, I did not include the existential quantifier in the definition of the language.

I commented at the time that a notion of explicit existential was needed for full generality. I suggest that such a notion may be better introduced by means of a predicate, like "runs" or "beats" which we have seen often in this document, than by using the conventional quantifier, $\exists$. One motivation for this is that predicates describe situations in the discourse world, and that the existence of an entity in the discourse world is certainly a situation therein. Contrast this with the essentially mathematical behaviour of the (conventional) universal quantifier, which affects the behaviour of a predicate in relation to the members of a set, without affecting the nature of that set itself; remembering that existence in this context means that such a set is inhabited.

Another argument for using a GRL predicate to represent strong existential statements is that such statements may be tensed and modal. It is not possible to represent this with the conventional quantifier without the addition of modal operators, which are unnecessary elsewhere, and, along the same lines as the argument above, this is not really the kind of information we want to associate with a quantifier anyway.

3.3. Binary Logical Connectives

The further (binary) connectives required in GRL are the four conventional syncategorematic operators, *and*, *inclusive or*, *exclusive or*, and *implies*, which could be written $\wedge \vee \oplus \Rightarrow$ respectively. These connectives apply to well-formed GRL expressions to produce other well-formed GRL expressions. I include exclusive or as a distinguished symbol because of the prevalence of exclusive disjunction in English – the word "or" rarely means inclusive or.

These new operators may be used to combine closed predicates (see below) or other logical combinations in the much same way as in those in FOPC. There is, however, an important difference. GRL is used in George to represent the discourse input, and not just its truth-functional semantics. Thus, the interpretation of the GRL connectives may be different

from the conventional interpretation of the corresponding connectives in FOPC, particularly in cases where the success or failure of reference affects the truth of an utterance. Such a difference in interpretation would be reflected in extensions to the translation algorithm specified in Chapter 4.

3.4. Negation

Negation in a representation system like GRL is an interesting problem. There are two distinct kinds of negation, one of which affects reference directly.

The less interesting kind is what might be called truth-functional negation. This is negation more or less exactly as in FOPC, as in sentences like 249. Note that the negation in this sentence does not affect the reference of the embedded noun phrases. Like the binary operators in Section 3.3, it is possible (though unlikely, here) that the semantic translation for this operator may yield something more complicated that just FOPC negation.

"Jim didn't beat his donkey yesterday." 249

This kind of negation could be written in the same way as in FOPC, by the addition of a unary negation operator, $\neg$, (*not*) applying to GRL or $GRL_{\exists}$ closed predicates.

The much more interesting form of negation is that where the negation affects the reference in a sentence. The strongest example of this is sentence 250a. To cover this form of negation, I propose the notion of *referential negation*, which could be denoted by a unary operator ¡ applying to $GRL_{\exists,\neg}$ references, giving the expression in 250b.

"No man beats the donkey." 250a

[beats,ref2¯donkey!,ref1¯man¡] 250b

There are two readings of this sentence. In one, where "No man" is referential, the meaning is "For all men x, x does not beat the donkey" where x ranges over the set of men referred to. The other is equivalent to "There does not exist a man who beats the donkey" which means something different from the first reading in some circumstances. A first approximation to the second reading would be that the "man" reference must have an empty candidate set; alternatively, it could be represented directly by negation of the existentials in $GRL_{\exists,\neg}$, which might involve an empty candidate set as well. This, however, does not capture the first reading, which would be directly representable in the

$GRL_{\neg}$. The form in 250b might then be regarded as an adaptable ambiguous representation, and be coerced to the correct specific reading when (if) possible.

When we introduce quantifiers and/or dependent references, the problem becomes harder. Consider the sentences in 251.

"No man beats a donkey." 251a

"Not all the men beat the donkey." 251b

In sentence a, the negation of the subject can introduce an explicit strong negated existential into the meaning of the sentence – to mean something like "forall men x, there does not exist a donkey which x beats.". This is representable in $GRL_{\exists,\neg}$. An alternative reading is where the phrase "a donkey" is introductory – consider the continuing sentence "The donkey is quite pleased about this.", which refers successfully. Again, then, we have an ambiguous form, which it may be appropriate to represent ambiguously in an extension of GRL. I suggest that rules like those governing set reference manipulation in George might well be definable to allow the effect of the ¡ operator proposed above to propagate in a way similar to indices.

Similarly, in sentence 251b, we would wish to define an operation to express the meaning of the sentence: "Some of the men beat the donkey and some do not.". This might be characterisable in terms of the entity token and index partition operations in conjunction with the ¡ operator, in much the same way as different set interactions are captured by the strong and weak index operations.

3.5. Interrogatives and Imperatives

Some interrogative references in $GRL_{\exists,\neg,¡}$ might be represented by a query operator, ?. This would apply to references in the same way as the definite operator, with an almost identical effect. George could then neatly capture Wh-questions, by the addition of extra behaviour to the higher level parts of the George system, as follows.

Such a queried reference would never be introductory, but would cause an attempt to find a set of candidate entity tokens as per definite references. The answer (the need for which would be detected at a higher level in the George system, from the presence of the ? operator) could then be given as the expression was added to the discourse memory (*ie* at the end of the sentence). The form of the answer, of course, would depend on the form of the reference (*ie* indexed, bounded, simple), and the content of its binding (no entity

tokens, an expected number of tokens or a token specifying a set of an expected size, or some unexpected result).

The other kind of interrogative, characterisable as questions about the predicates describing discourse situation could be captured in the Discourse Memory by the addition of further predicate modifier symbols to the language, again with a detection machanism at the higher level to allow questions to be answered.

This approach would also work for imperatives, with some form of action simulation mechanism (maybe like SHRDLU's robot) instead of the question answering mechanism.

3.6. Extending #-Abstraction

The current form of #-abstraction is rather limited, and, in fact, not perfect for the purposes of early reference analysis. In particular, as I explained in Chapter 6, George does not produce context extensions in a form suited to comparison with the existing discourse until all the argument positions in them are at least partially filled, and GRL is unable to express selectional constraints on verb positions.

The reason for this is that verb positions are filled by λ-reduction, and not #-reduction. In GRL, partial noun phrases are represented by #-abstractions, and partial verb phrases by λ-abstractions. The two do not interact.

A better solution would be to represent the verbs as #-abstractions, with the reference symbols already built in to them, so the lexical entry for "runs" would look like this:

$$\text{runs} := s\backslash np : ref1\#[runs, ref1\ \hat{}\ runner] \qquad 252$$

where "runner" is a compound sort restricting the reference to refer to something consistent with "river" or something consistent with "[has legs, is animate]" for example. (Formalisms for such compound sort manipulation, which fit in with the style of the George taxonomy, already exist – *eg* KL-ONE [Brachman & Smolze, 1985].)

The composition of noun phrases could then proceed in the same way in as the current George system, viewed as agglommeration of noun phrase properties on some basic reference symbol. This information would then be passed to the reference symbol in the verb on arrival of the verb at the input, by sorted unification of the existing reference and the reference in the verb translation (see 252). (The case where unification would be required rather than, say, sort application, is detectable directly from the syntax of GRL.)

The unification algorithm would require consistency between the sort information already in the verb and that in the noun phrase. This would allow application of selectional constraints, as in 252.

This approach is more suited to context extension comparison than the λ-abstracted version because that comparison is defined over expressions containing references and not GRL variables. The existing matching operation and subsequent inference would proceed correctly on such expressions without modification.

Note that, even with this extension, GRL will still need λ-abstraction to allow for manipulations of discourse such as that already used in the construction of context extensions (see Chapter 5) or proposed in Section 4 of this chapter.

3.7. Summary

In this section I have proposed the addition of operators and predicates to GRL as presented in Chapter 4, to allow representation of the conventional binary logical connectives, negation (both truth-functionally and referentially), interrogatives, and imperatives. I have sketched out how they might affect the behaviour of an extended George system.

I have also proposed a necessary extension to the idea of #-abstraction to allow a more incremental analysis of the reference of context extended expressions, and to admit the expression of selectional constraints.

4. Meta-Linguistic Functions

4.1. Introduction

It is a basic principle of the George Representation Language that linguistic effects, particularly those which affect reference, should be explicitly represented. This requires us to consider various special forms of reference which must, therefore, be represented specially. This has the advantage that all the information in the discourse is always available to us in making inferences about the meaning conveyed by the discourse. The particular forms I will discuss here are reflexives and what [Webber, 1979] calls "one"-anaphora, though there are many others.

I class these phenomena under the heading "Meta-Linguistic Functions" because they are references whose meaning is defined in terms of other references – in a sense, for example, the meaning of a reflexive reference is that it is a reference to the same thing as the actor reference of the clause in which it appears.

4.2. Reflexive Pronouns

The correct way to implement reflexives in George is an open question. The most obvious method, duplication of the reference co-referential with the reflexive by the parser, by use of a higher-order or meta-level translation of the reflexive, is analogous with Steedman's **W** combinator solution, where the argument to a function is replicated, effectively by application of the λ-term ($\lambda f.\lambda a.faa$). It would involve the introduction of meta-terms into the George Representation Language, which is undesirable because of the theoretical complications involved.

Given the nature of reflexives and the information they convey, it seems likely that any treatment involving special manipulation of the discourse translation to duplicate references will involve higher-order operations as part of composition. Such treatment might therefore be considered undesirable, because of the complexity of those operations.

Perhaps the best solution, then, is to use a special GRL notation for reflexives, and let the DeReferencer handle the referential analysis – which is after all its function – as a special case. This is in keeping, too, with the idea of representing (something close to) surface form in GRL, with a special symbol for each linguistic feature.

4.3. "One"-Anaphora

[Webber, 1979] gives a method of resolving "one"-anaphora, as seen in "There were two t-shirts; I liked the blue one.". Webber requires that the head noun of a noun phrase be given a special status in her representation, because (she claims) it is necessary to identify the head noun in order to deal with one-anaphora. [Webber, 1979, page 3-13] gives translations of sorts like this:

"T-shirt"	as T-shirt (short for λ(u:T-shirt)[True])	
"cotton T-shirt"	as λ(u:T-shirt)[Cotton u]	
"T-shirt that Wendy gave Fred"	as λ(u:T-shirt)[Gave Wendy, Fred, u]	253

The representation distinguishes the head noun ("T-shirt") from any modifiers applied to it, as well as separating the information contained in the determiner (not represented here) from that in the nominal.

However, this separation of the head noun is not generally adequate for the purposes on account of which it is justified. Consider, for example, the discourse in 254.

"There are three donkeys in a field, two brown and one black.
The cruel farmer beats the large brown donkey. He also mistreats the small one." 254

One possible reading of this discourse is that the "one"-anaphor in the final noun phrase refers to the entity described by "brown donkey". As such, Webber's representation is inadequate, since she does not distinguish between adjectives in the same way as between adjectives and nouns. I suggest that what is really needed here is a pragmatic view like that which I applied to the dereferencing of context extended references in Chapter 6.

A first approximation to the behaviour of the non-introductory "one"-anaphor (call it O) is as follows. There must be a non-singleton set, S, of entities which have some common sorts and some differentiating sorts. This set, or a subset of it, must be bound to a reference, R, recently preceding O, but not in the same clause, whose sort does not subsume that of O. O must then be bound to the entity, if one exists, which is selected uniquely by taking the intersection of the candidate set of O treated as an ordinary reference and the candidate set of a reference R' where R' is R with any information contradicting the information in the "one"-anaphor removed.

To clarify this, consider 254. To analyse the "one"-anaphor, we proceed as follows. The candidate set of the anaphor treated as an ordinary reference contains the black donkey and the non-large brown donkey, but not the farmer, which is the referent of "He" in the same clause, or the other (large) donkey, which is inconsistent with the sort of the reference. The non-singleton set, S, is the set of brown donkeys, one of which is known to be large. The reference R is the "large brown donkey". R' is therefore a reference to a "brown donkey" ("large" is removed because it is inconsistent with "small") – its candidate set is therefore both brown donkeys. The intersection of the two candidate sets leaves us with the non-large brown donkey, which can now be marked as small.

The method also works in a degenerate sense when there are no explicit contradictions between the reference R and the anaphor O. If the "one"-anaphor had been "the old one" for example, the set S would simply have been the three donkeys, with the candidate set of O being the same until some donkeys were found not to be old.

The method is based on the same assumption as Webber's, that some reference is needed to the surface form of the preceding discourse, but the view is more general, in that it is not limited to head nouns.

5. The Discourse Entity Level and (Co-)Specification

All the manipulations demonstrated in this document operate on discourse memory entries, entity tokens, and/or bindings. However, I mentioned in Chapter 3 that we need a further theoretical level of reference information beyond the entity token level – that of the *discourse entity* – and that an entity token and the number and sort information associated with it by its defining reference(s) may be viewed as *specifying* the (possibly singleton) sets of discourse entities with which it is related.

This is because the entity tokens are in a fairly strong sense intensional, and the specification relation, above, is rather like extension of the tokens. In particular, this further level is needed so that we can distinguish between utterances which refer to "real" entities in the discourse world, specified by entity tokens, and utterances which refer only to intensional entities, or, if one prefers, prototypes. A good example of this latter kind of utterance is generic reference. Consider the discourses in 255.

"A man beats a donkey.
A donkey is a faithful animal.
He still loves the man who beats him." 255

It is possible to treat the reference in this examples exactly as the reference in Chapter 6 was treated, regardless of whether the utterances are generic or not; "He" in the third sentence is simply either bound to the "real" donkey entity token or to the generic one, which can be expressed in the usual George-way. The difference comes only when we try to follow inference through the discourse. The two possible sequences of inference are based on the two possible readings of "He", above.

Where "He" refers to the "real" donkey, the inference is as follows: x beats y. All s in S are faithful. x is in S. Therefore x is faithful. Therefore x loves y.

Where "He" refer to the generic, the inference runs this way: x beats y. All s in S are faithful. Therefore, all s in S love f(s) for some f. x is in S. Therefore, x loves f(x), therefore x loves y.

The interesting point here is the inference that the predication(s) applied to the generic also apply to the "real" donkey. This fact needs to be represented explicitly in George. To do so, and assuming (rather freely, since this is a hard question) that we can detect generics, I propose the idea of *cospecification*, under which two entity tokens may specify (some of) the same discourse entities by virtue of their own (the entity tokens') equivalence (in some sense). In general, we might say that the generic entity token cospecifies with any specific entity token with which it is consistent by sort.

Thus, the entity token of the generic is indirectly associated with the discourse entity of the ordinary introductory reference. This coincides with the intuition that generics refer to individuals merely by virtue of the individuals' explicit introduction into the discourse in which the generic occurs. More formally, it is useful to take the view that cospecification is actually a mapping from generic entity tokens to specific entity tokens, because this view makes explicit the difference between the introductory entity token's specifying its own discourse entity and the generic's specifying a discourse entity introduced with a different entity token.

Finally, it is worth pointing out that the view taken in George as presented here is consistent with the notion of adaptability: the entity token behaviour is the same whether references and entities are individual or generic.

6. Summary

In this chapter, I have sketched proposals for the extension of the George system and theory in various ways. This intention has been to give a flavour of possible extensions, rather than full detail.

I have covered extensions of GRL for representation of explicit strong existentials, compound expressions such as conjunctions and disjunctions, negation, interrogatives and imparatives. I have suggested an improvement of the #-abstraction mechanism defined in Chapter 4. I have suggested a view of reflexives and "one-anaphora" which places them in a category of references to or about references, and, finally, I have sketched a possible approach to intension references such as generics.

7. Afterword

This discussion of possible extensions completes my presentation of the George Parsing and DeReferencing System and the theory behind it. My hope is that at least some of these extensions will be investigated further and implemented in the not-too-distant future.

The work presented here constitutes a fairly radical step outside computational linguistic convention. As such, there are bound to be large numbers of open issues. I have tried in this document to give a good feel for the solutions to an interesting and wide-ranging selection of phenomena within the George framework.

I hope I have succeeded.

Bibliography

Ajdukiewicz, K — **1935** Die syntaktische Konnexität
in "Polish Logic", Oxford University Press, 1967

Alshawi, H — **1987** Memory and Context for Language Interpretation
Cambridge University Press

Altmann, G — **1987** On being led up the Garden Path
School of Epistemics, University of Edinburgh

Altmann, G — **1986** Reference and Resolution of Local Ambiguity: Interaction in Human Sentence Processing
PhD Thesis, University of Edinburgh

Barwise, J; **Perry**, J — **1983** Situations and Attitudes
Bradford Books

Bobrow, R; **Webber**, B — **1980** PSI-KLONE: Parsing and Semantic Interpretation in the BBN Natural Language Understanding System
in "Proceedings of the Canadian Society for Computational Study of Intelligence", pp 131-142

Bobrow, R; **Webber**, B — **1980** Knowledge Representation for Syntactic/Semantic Processing
in "Proceedings of the Canadian Society for Computational Study of Intelligence", pp 131-142

Brachman, R; **Smolze**, J — **1985** An Overview of the KL-ONE Knowledge Representation System
Cogitive Science, #9, pp 171-216

Brailsford, D; **Walker**, A — **1979** Introductory Algol 68 Programming
Ellis-Horwood

Bundy, A; **Byrd**, L; **Luger**, G; **Mellish**, C; **Palmer**, M — **1979** Solving Mechanics Problems Using Meta-Level Inference
in "Proceedings of the 6th International Joint Conference on Artificial Intelligence", Tokyo, Japan

Burt, A; **Hill**, P; **Lloyd**, J — **1990** Preliminary Report on the Programming Language Gödel
Technical Report TR-90-02, Department of Computer Science, University of Bristol

Carter, D **1988** A Shallow Processing Approach to Anaphor Resolution
PhD Thesis, Technical Report No 88, University of Cambridge Computer Laboratory, Cambridge

Calder, J; **Klein**, E; **Zeevat**, H **1988** Unification Categorial Grammar: A Concise, Extendable Grammar for Natural Language Processing
in "Proceedings of the 12th International Conference on Computational Linguistics", Budapest

Charniak, E **1972** Towards a Model of Children's Story Comprehension
PhD Thesis, MIT AI TR 266

Chomsky, N **1965** Aspects of the Theory of Syntax
MIT Press, Cambridge, MA

Church, A **1940** A Formulation of a Simple Theory of Types
Journal of Symbolic Logic, #5, pp 56-68

Church, A **1941** The Calculi of Lambda Conversion
Princeton University Press; reprinted in 1963 by University Microfilms Inc., Ann Arbor, Michigan, USA

Clocksin, W; **Mellish**, C **1981** Programming in Prolog
Springer Verlag

Crain, S; **Steedman**, M **1985** On Not Being Led Up The Garden Path: The Use of Context in the Psychological Parser
in "Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives", eds. Dowty, Kartunnen & Zwicky

Dowty, D; **Wall**, R; **Peters**, S **1981** Introduction to Montague Semantics
Synthese Series, D Reidel

Dowty, D **1985** Type-Raising, Functional Composition, and Non-Constitutent Conjunction
in "Categorial Grammars and Natural Language Structures", eds. Öhrle, Bach & Wheeler; D Reidel

Fine, C **1985** Reasoning with Arbitrary Objects
Aristotelian Society Series, Volume 3, Basil Blackwell, Oxford

Gazdar, G; **Klein**, E; **Pullum**, G; **Sag**, I **1985** Generalised Phrase Structure Grammar
Basil Blackwell, London

Grosz, B **1977** The Representation and Use of Focus in Dialogue Understanding
Technical Note 151, SRI International, Menlo Park, California

Haddock, N **1985** Computing Noun Phrase Reference
Working Paper 182, Department of Artificial Intelligence, University of Edinburgh

Haddock, N **1987** Incremental Parsing and Combinatory Categorial Grammar
in "Edinburgh Working Papers in Cognitive Science", Vol 1, eds. Haddock, Klein & Calder, Centre for Cognitive Science, University of Edinburgh

Haddock, N **1989** Incremental Semantics and Interactive Syntactic Processing
PhD Thesis, University of Edinburgh

van Harmelen, F; **Simpson**, A; **Giunchilia**,F; **Serafini**, L; **Smaill**, A **1990** A Discussion about Naming Relations
Report RFL/UvA/I.4/1, ESPRIT/BR Action P3178

Halliday, M **1967** Notes on Transitivity and Theme in English: II
Journal of Linguistics #3:199

Hirst, G **1981** Anaphora in Natural Language Understanding: A Survey
Lecture Notes in Computer Science, #119, Springer-Verlag

Hobbs, J; **Schieber**, S **1981** An Algorithm for Generating Quantifier Scopings
"Computation Linguistics" #13, numbers 1-2, January-June 1987

Johnson, M; **Klein**, E **1986** Discourse, Anaphora and Parsing
"Proceedings of the 11th International Conference on Computational Linguistics", Bonn, 1986

Kamp, H **1981** A Theory of Truth and Semantic Representation
in "Formal Methods in the Study of Language", Vol. 136, eds. Groenendijk, Janssen & Stokhof; Amsterdam: Mathematical Centre Tracts

Kowalski, R **1979** Logic for Problem Solving
North-Holland Elsevier

Kim, J; **Kowalski**, R **1990** An Application of Amalgamated Logic to Multi-Agent Belief
in Proceedings, META-90, Leuven, Belgium, ed. M Bruynooghe

Lambek, J **1958** The Mathematics of Sentence Structure
American Mathematical Monthly, #65, pp 154-170

Lewin, I **1990** A Quantifier Scoping Algorithm without a Free Variable Constraint
in Proceedings, CoLing-90, Helsinki, Finland

Lloyd, J **1987** Foundations of Logic Programming
Symbolic Computation Series, Springer, Berlin

Marcus, M **1980** A Theory of Syntactic Recognition for Natural Language
MIT Press

Marslen-Wilson, W **1975** Sentence Perception as an Interactive Parallel Process
"Science", #189, pp 226-228

Marslen-Wilson, W; **Tyler**, L **1980** The Temporal Structure of Spoken Language Understanding
"Cognition", #8, pp 1-74

Marslen-Wilson, W **1987** Functional Parallelism in Spoken Word Recognition
"Cognition", #25, pp 71-102

Mellish, C **1981** Coping with Uncertainty: Noun Phrase Interpretation and Early Semantic Analysis
PhD Thesis, University of Edinburgh

Mellish, C **1982** Incremental Evaluation: An Approach to the Semantic Interpretation of Noun Phrases
Ellis-Horwood

Mellish, C **1984** Incremental Semantic Interpretation
in "Parsing Natural Language", eds. Sparck-Jones & Wilks, Ellis-Horwood

Mellish, C **1984** Computer Interpretation of Natural Language Descriptions
Ellis-Horwood

Montague, R **1973** The Proper Treatment of Quantifiers in Ordinary English
in "Approaches to Natural Language", eds. Hintikka, Moravcsik & Suppes; D Reidel

Moortgat, M **1988** Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus
Foris Publications

Pareschi, R **1987** Combinatory Grammar, Logic Programming, and Natural Language Processing
in "Edinburgh Working Papers in Cognitive Science", Vol 1, eds. Haddock, Klein & Calder; Centre for Cognitive Science, University of Edinburgh

Pareschi, R **1988** Chart Parsing Categorial Grammars
PhD Thesis, University of Edinburgh

Ritchie, G **1988** Problems in Local Semantic Processing
in "Proceedings of the AISB Conference", ed. Brady, University of Edinburgh

Shillcock, R **1982** The On-line Resolution of Pronominal Anaphora
Language and Speech, #25, pp 385-402

Sidner, C **1979** Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse
PhD Thesis, Departemnt of Electrical Engineering and Computer Science, MIT

Sidner, C **1983** Focussing in the Comprehension of Definite Anaphora
in "Computational Models of Discourse", eds. M Brady, R C Berwick; MIT Press

Steedman, M **1985** Dependency and Coordination in the Grammar of Dutch and English
Language, Vol. 61 pp 523-568

Steedman, M **1987** Combinatory Grammars and Parasitic Gaps
in "Edinburgh Working Papers in Cognitive Science", Vol 1, eds. Haddock, Klein & Calder Centre for Coginitive Science, University of Edinburgh

Tyler, L; **Marslen-Wilson**, W **1977** The On-line Effects of Semantic Context on Syntactic Processing
Journal of Verbal Learning and Verbal Behaviour, Vol. 16 pp 683-692

VanLehn, K **1978** Determining the Scope of English Quantifiers
Technical Report TR-483, AI Lab, MIT, Stanford

Waltz, D **1972** Generating Semantic Descriptions from Drawings of Scenes with Shadows
Technical Report, MIT

Webber, B **1979** A Computational Approach to Discourse Anaphora
Harvard University Press

Webber, B **1983** So what can we talk about now?
in "Computational Models of Discourse", eds. M Brady, R C Berwick; MIT Press

Winograd, T **1971** Language as a Cognitive Process, Vol 1
Harvard University Press

Winograd, T **1972** Understanding Natural Language
Edinburgh University Press

Winograd, T **1973** A Procedural Model of Language Understanding
in "Computer Models of Thought and Language", eds. R C Schank, K M Colby; publ. Freeman

Wilks, Y **1975** A Preferential Pattern-seeking Semantics for Natural Language Inference
Artificial Itelligence #6 pp53-74

Wilks, Y; **Huang**, X; **Fass**, D **1985** Syntax, Preference and Right Attachment
in "Proceedings of the 9th International Joint Conference on Artificial Itelligence", pp779-784

Woods, W **1970** Transition Network Grammars for Natural Language Analysis
Communications of the ACM, Vol 13, pp 591-606

Zeevat, H; **Klein**, E; **Calder**, J **1986** Unification Categorial Grammar
in "Edinburgh Working Papers in Cognitive Science", Vol 1, eds. Haddock, Klein & Calder; Centre for Coginitive Science, University of Edinburgh

Appendix A
Glossary of Terms

Abstraction

The removal of an GRL object from an expression, coupled with the annotation of that part of the expression as lacking information. GRL has two forms of abstraction: # abstraction for references and λ abstraction for other GRL objects.

Adaptability

The property of a representation system which allows it to represent partial and/or ambiguous information in such a way that completing and/or disambiguating transformations can be (subsequently) applied.

Application

The association of a function with an argument, prior to evaluation.

Binding

A relation between a set of co-referential references and a set of entity tokens (the candidate set of the reference(s)), expressing that the referring expressions represented by the references refer to the discourse entities specified by the entity tokens.

Bounding Constraint

An arithmetic relation between the upper bounds the indices of two or more GRL indexed references.

Candidate Set

A collection of entity tokens each of which is consistent in number and sort with a reference.

Coercion

A transformation, applied to the syntax and semantics of a partial translation in GRL, and guided by the syntax (and sometimes semantics) of a newly input word. Coercions enable disambiguation of adaptable ambiguous expressions.

Composition

The association of one function with another prior to evaluation.

Consistency

The property of two GRL references of being able to be bound to the same entity tokens. Characterisable by number consistency rules and sort consistency defined by movement around a sort hierarchy.

Context Extension

That part of a GRL expression or reference which arises from subordinate noun phrase modification.

Coreference

Reference by two or more GRL references to the same entity token(s).

Definiteness

The property of GRL references, denoted by the ! operator, which requires them to be non-introductory, if possible. Ultimately, definiteness requires uniqueness of reference in the singular case and totality of reference across a set in the plural.

DeReferencing

The process of finding and constraining the candidate set of a GRL reference.

Discourse Entity

The object with which George would represent the actual things in the discourse world. Discourse entities are of the same form as entity tokens; it is only the purpose they serve that is different. Discourse entities are related to entity tokens by the *specification* mapping, qv.

Discourse State

The state of the George system after processing any particular input utterance. Composed of Discourse Memory, Bindings, Bounding Constraints and Entity Tokens.

Discourse World

The world in which the discourse is set. May or may not coincide with the real world.

Early

The property of a language analysis system of incorporating information received at

the input into the representation of the discourse sooner than is conventional, in particular before the end of a sentence.

Entity Token

The entity token is the point at which the constraint information gleaned from the various references within a discourse to a particular entity collects. It may be viewed as something like the sense of the entity or entities to which reference is being made.

Evaluation

Manipulation of GRL expressions after application and composition into a canonical form.

GRL

The George Representation Language.

Incremental

The property of a language analyser that allows information to be added piecemeal. The representation system used must be adaptable.

Incremental Parsing

Language analysis word-by-word.

Index

A GRL variable ranging over $\mathbb{N}$, sometimes with an upper bound. Applied to GRL references by the strong and weak index operators, $\otimes$ and $\times$ respectively. Used to represent and reason about the relations expressed by sentence containing set references.

Index Application

The application of GRL index to a reference, with one of the index application operators, $\otimes$ and $\times$.

Index Expansion

The rewriting of expressions containing an index with upper bound N by N copies, each with a reference to a member of a set of N members referred to by the expression before rewriting.

Index Propagation

Movement of indices between references according to certain rules. Part of the George process of reasoning about set reference.

Number

The size of a set.

Precursor

A GRL expression in the Discourse Memory which a context extension may match.

Predicate

A relation between entities in the discourse world.

Predicate Symbol

The name of a predicate.

Protraction

The George operation allowing strictly incremental parsing where constituents would normally be stacked in a non-incremental system.

Quantification

Strong Quantification is the application of predicates to sets in such a way that the predication is exclusively of the individuals and not of the set as a whole. Examples: Each, Every.

Weak Quantification is the application of predicates to sets so that predication may be of the individuals or of the collective set. Example: Some, The (both plural).

Quantifier Expansion

The GRL operation of textually moving quantifiers embedded in an expression to the front of that expression.

Reduction

Part of the evaluation operation. An abstracted expression and an argument of the right type may be reduced to a new, less abstracted expression.

Refinement

The addition of information to GRL references and George Bindings and Entity Tokens. Often results in the constraining of candidate sets.

Relevance

The property of a precursor of being a possible match for a context extension.

Reference

The representation in GRL of a referring expression in English. NB: not the same as the reference of a noun phrase in the common linguistic sense.

Situation Information

Information defining relationships in the discourse world between the entities taking part in a discourse. The boundary between this and sort information is fuzzy.

Sort

Information defining the basic properties of the entity tokens available as candidates for binding to a GRL reference.

Sort Association

Association of sort information with a GRL reference symbol or George Entity token.

Specification

The relation between George entity tokens and the discourse entities in the discourse world.

Topic

The important reference in a context extension; corresponds with the gap in a relative clause.

Underspecification

The property of being incompletely defined. Mostly applied to set reference here.

Appendix B
Program Execution Traces

1. The Simplest Case

```
?- listen.
 ** George parser **

State 0:



::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

>> A
================================================
Next word: A
------------------------------------------------

     np(sing,_)/nmod(sing,_)/n(sing):
        refl#λvar1.[var1,refl] ▸ refl

refl is bound to e1

>> man
================================================
Next word: man
------------------------------------------------

     np(sing,_)/nmod(sing,_)/n(sing):
        refl#λvar1.[var1,refl] ▸refl
   + n(sing):man
  → np(sing,_)/nmod(sing,_):
        λvar1.[var1,refl~man] ▸ refl~man

refl is bound to e1~man
```

```
>> who
================================================
Next word: who
------------------------------------------------

     np(sing,_)/nmod(sing,rel):λvar1.[var1,refl~man] ▸ refl~man
   + nmod(sing,rel)/(s/np(sing,nom)):λvar2.var2
   → np(sing,_)/(s/np(sing,nom)):
        λvar2.[var2,refl~man] ▸ refl~man

refl is bound to el~man

------------------------------------------------
     np(sing,_)/nmod(sing,rel):λvar1.[var1,refl~man] ▸ refl~man
   + nmod(sing,rel)/(s/np(_,nom)/np(sing,acc))/np(_,nom):
        λvar3.λvar4.[var4,var3]
   → np(sing,_)/(s/np(_,nom)/np(sing,acc))/np(_,nom):
        λvar3.λvar4.[[var4,var3],refl~man] ▸ refl~man)

refl is bound to el~man

>> is
================================================
Next word: is
------------------------------------------------

     np(sing,_)/(s/np(sing,nom)):λvar2.[var2,refl~man] ▸ refl~man
   + s/np(sing,nom)/np(sing,nom):λvar10.λvar11.coref(var10,var11)
   → np(sing,_)/np(sing,nom):
        λvar10.coref(var10,refl~man) ▸ refl~man

refl is bound to el~man

------------------------------------------------

     np(sing,_)/(s/np(sing,nom)):λvar2.[var2,refl~man] ▸ refl~man
   + s/np(sing,nom)/(n(_)/n(_)):ref2#λvar12.coref(ref2,var12))
   → np(sing,_)/(n(_)/n(_)):
        ref2#coref(ref2,refl~man) ▸ refl~man

ref2 and refl~are bound to el~man

------------------------------------------------

     np(sing,_)/(s/np(sing,nom)):λvar2.[var2,refl~man] ▸ refl~man
   + s/np(sing,nom)/part(pres,pres,cont,act,indic):
        λvar13.λvar14.[var13,var14]
   → np(sing,_)/part(pres,pres,cont,act,indic):
        λvar13.[var13,refl~man] ▸ refl~man

refl is bound to el~man
```

```
>> unkind
================================================
Next word: unkind
------------------------------------------------

     np(sing,_)/(n(_)/n(_)):
        ref2#coref(ref2,ref1~man) ▸ ref1~man
   + n(_)/n(_):unkind
  → np(sing,_):coref(ref2~unkind,ref1~man) ▸ ref1~man

ref2 and ref1 are bound to e1~man~unkind

>> beats
================================================
Next word: beats
------------------------------------------------

     np(sing,nom):coref(ref2~unkind,ref1~man) ▸ ref1~man
   + s/np(sing,nom)/np(_,acc):
        λvar15.λvar16.[beat(pres,pres,perf,act,indic),var15,var16])
  → s/np(_,acc):
        λvar15.coref(ref2~unkind,ref1~man) ▸
                 [beat(pres,pres,perf,act,indic),var15,ref1~man]

ref2 and ref1 are bound to e1~man~unkind

>> a
================================================
Next word: a
------------------------------------------------

     s/np(sing,acc):
        λvar15.coref(ref2~unkind,ref1~man) ▸
                 [beat(pres,pres,perf,act,indic),var15,ref1~man]
   + np(sing,acc)/nmod(sing,_)/n(sing):
        ref3#λvar17.[var17,ref3] ▸ ref3
  → s/nmod(sing,_)/n(sing):
        ref3#λvar17.[var17,ref3] ▸
                       coref(ref2~unkind,ref1~man) ▸

[beat(pres,pres,perf,act,indic),ref3,ref1~man]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2
```

```
>> donkey.
================================================
Next word: donkey
------------------------------------------------

     s/nmod(sing,_)/n(sing):
        ref3#λvar17.[var17,ref3] ▸
                        coref(ref2~unkind,ref1~man) ▸

[beat(pres,pres,perf,act,indic),ref3,ref1~man]
   + n(sing):donkey
  → s/nmod(sing,_):
        λvar17.[var17,ref3~donkey] ▸
                  coref(ref2~unkind,ref1~man) ▸

[beat(pres,pres,perf,act,indic),ref3~donkey,ref1~man]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey

================================================
End of Sentence

  ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),ref3~donkey,ref1~man]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey

------------------------------------------------------------------
∀x.(donkey(x) ⇒ nature(x,animal)) ∧
∀x.(man(x) ⇒ gender(x,masculine)) ∧ ∀x.(man(x) ⇒ type(x,human)) ∧
∀x.(man(x) ⇒ nature(x,animal)) ∧
∃v1.(donkey(v1) ∧ ∃v2.(man(v2) ∧ unkind(v2) ∧ beat(v1,v2)))
------------------------------------------------------------------

>> ^D !! End of input Stream!!

 ** Terminating parse **

yes
```

2. Simple Definite Reference

```
?- listen.
  ** George parser **

State 1:

  ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),ref3~donkey~brown,ref1~man]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

>> A
================================================
Next word: A
------------------------------------------------

     np(sing,_)/nmod(sing,_)/n(sing):
        ref4#λvar18.[var18,ref4] ▸ ref4

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~brown
ref4 is bound to e3

>> woman
================================================
Next word: woman
------------------------------------------------

     np(sing,_)/nmod(sing,_)/n(sing):
        ref4#λvar18.[var18,ref4] ▸ ref4
   + n(sing):woman
   → np(sing,_)/nmod(sing,_):
        λvar18.[var18,ref4~woman] ▸ ref4~woman

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~brown
ref4 is bound to e3~woman
```

```
>> feeds
================================================
Next word: feeds
------------------------------------------------

     np(sing,nom)/nmod(sing,_):
        λvar18.[var18,ref4~woman] ▸ ref4~woman
   + s\np(sing,nom)/np(_,acc):
        λvar19.λvar20.[feed(pres,pres,perf,act,indic),var19,var20]
  → s/np(_,acc):
        λvar19.[feed(pres,pres,perf,act,indic),var19,ref4~woman]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~brown
ref4 is bound to e3~woman

>> the
================================================
Next word: the
------------------------------------------------

     s/np(sing,acc):
        λvar19.[feed(pres,pres,perf,act,indic),var19,ref4~woman]
   + np(sing,acc)/nmod(sing,_)/n(sing):
        ref5#λvar23.[var23,ref5!] ▸ ref5!
  → s/nmod(sing,_129688)/n(sing):
        ref5#λvar23.[var23,ref5!] ▸

[feed(pres,pres,perf,act,indic),ref5!,ref4~woman]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~brown
ref4 is bound to e3~woman
ref5 may be bound to e2~donkey~brown
       or to e1~man~unkind

------------------------------------------------

     s/np(plur,acc):
        λvar19.[feed(pres,pres,perf,act,indic),var19,ref4~woman]
   + np(plur,acc)/nmod(plur,_)/n(plur):
        ref6#λvar24.∀ind1.[var24,ref6!×ind1] ▸ ref6!×ind1
  → s/nmod(plur,_)/n(plur):
        ref6#λvar24.∀ind1.[var24,ref6!×ind1] ▸
                          [feed(pres,pres,perf,act,indic),
                             ref6!×ind1,
                             ref4~woman]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~brown
ref4 is bound to e3~woman
ref6 is bound to e2~donkey~brown
       and e1~man~unkind
```

```
>> donkey.
================================================
Next word: donkey
------------------------------------------------

     s/nmod(sing,_)/n(sing):
        ref5#λvar23.[var23,ref5!] ▸

[feed(pres,pres,perf,act,indic),ref5!,ref4~woman]
   + n(sing):donkey
  → s/nmod(sing,_):
        λvar23.[var23,ref5~donkey!] ▸

[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey

================================================
End of Sentence

  ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),ref3~donkey~brown,ref1~man]
  ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey

------------------------------------------------------------------
∀x.(donkey(x) ⇒ nature(x,animal)) ∧
∀x.(man(x) ⇒ gender(x,masculine)) ∧ ∀x.(man(x) ⇒ type(x,human)) ∧
∀x.(man(x) ⇒ nature(x,animal)) ∧
∀x.(woman(x) ⇒ gender(x,feminine)) ∧
∀x.(woman(x) ⇒ type(x,human)) ∧ ∀x.(woman(x) ⇒ nature(x,animal)) ∧
∃v1.(woman(v1) ∧ ∃v2.(donkey(v2) ∧ ∃v3.(man(v3) ∧
     unkind(v3) ∧ beat(v2,v3) ∧ feed(v2,v1))))
------------------------------------------------------------------

>> ^D!! End of input Stream!!

 ** Terminating parse **

yes
```

3. Definite Post-Modified Reference

```
| ?- listen.
** George parser **

State 1:

  ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),ref3~donkey~brown,ref1~man]
  ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

>> He
================================================
Next word: He
------------------------------------------------

     np(sing,nom)/nmod(sing,_):
        λvar25.[var25,ref7~male!] ▸ ref7~male!

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey
ref7 may be bound to e2~brown~donkey
                     (if e2 is consistent with [male])
       or to e1~man~unkind

>> hates
================================================
Next word: hates
------------------------------------------------

     np(sing,nom)/nmod(sing,_):
        λvar25.[var25,ref7~male!] ▸ ref7~male!
   + s\np(sing,nom)/np(_,acc):
        λvar26.λvar27.[hate(pres,pres,perf,act,indic),var26,var27]
  → s/np(_,acc):
        λvar26.[hate(pres,pres,perf,act,indic),var26,ref7~male!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey
ref7 may be bound to e2~brown~donkey
                     (if e2 is consistent with [male])
       or to e1~man~unkind
```

```
>> the
================================================
Next word: the
------------------------------------------------

     s/np(sing,acc):
        λvar26.[hate(pres,pres,perf,act,indic),var26,ref7~male!]
   + np(sing,acc)/nmod(sing,_)/n(sing):
        ref8#λvar30.[var30,ref8!] ▸ ref8!
  → s/nmod(sing,_)/n(sing):
        ref8#λvar30.[var30,ref8!] ▸

[hate(pres,pres,perf,act,indic),ref8!,ref7~male!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey
ref7 may be bound to e2~brown~donkey
                    (if e2 is consistent with [male])
       or to e1~man~unkind
ref8 may be bound to e3~woman
       or to e2~brown~donkey
       or to e1~man~unkind

------------------------------------------------

     s/np(plur,acc):
        λvar26.[hate(pres,pres,perf,act,indic),var26,ref7~male!]
   + np(plur,acc)/nmod(plur,_)/n(plur):
        ref9#λvar31.∀ind2.[var31,ref9!×ind2] ▸ ref9!×ind2
  → s/nmod(plur,_)/n(plur):
        ref9#λvar31.∀ind2.[var31,ref9!×ind2] ▸
                              [hate(pres,pres,perf,act,indic),
                                 ref9!×ind2,
                                 ref7~male!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey
ref7 may be bound to e2~brown~donkey
                    (if e2 is consistent with [male])
       or to e1~man~unkind
ref9 is bound to e3~woman
       and e2~brown~donkey
       and e1~man~unkind
```

```
>> person
================================================
Next word: person
------------------------------------------------

     s/nmod(sing,_)/n(sing):
        ref8#λvar30.[var30,ref8!] ▸

[hate(pres,pres,perf,act,indic),ref8!,ref7~male!]
   + n(sing):person
   → s/nmod(sing,_):
        λvar30.[var30,ref8~person!] ▸
           [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey
ref7 may be bound to e2~brown~donkey
                    (if e2 is consistent with [male])
       or to e1~man~unkind
ref8 may be bound to e3~woman
       or to e1~man~unkind
```

```
>> who
================================================
Next word: who
------------------------------------------------

     s/nmod(sing,rel):
        λvar30.[var30,ref8~person!] ▸
           [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]
   + nmod(sing,rel)/(s\np(sing,nom)):lambda(var32,var32)
  → s/(s\np(sing,nom)):
        λvar32.[var32,ref8~person!] ▸
           [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey
ref7 may be bound to e2~brown~donkey (if e2 is consistent with
[male])
       or to e1~man~unkind
ref8 may be bound to e3~woman
       or to e1~man~unkind

------------------------------------------------

     s/nmod(sing,rel):
        λvar30.[var30,ref8~person!] ▸

[hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]
   + nmod(sing,rel)/(s\np(_,nom)/np(sing,acc))/np(_,nom):
        λvar33.λvar34.[var34,var33]
  → s/(s\np(_,nom)/np(sing,acc))/np(_,nom):
        λvar33.λvar34.[[var34,var33],ref8~person!] ▸
                          [hate(pres,pres,perf,act,indic),
                             ref8~person!,
                             ref7~male!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey
ref7 may be bound to e2~brown~donkey
                    (if e2 is consistent with [male])
       or to e1~man~unkind
ref8 may be bound to e3~woman
       or to e1~man~unkind
```

```
>> beats
================================================
Next word: beats
------------------------------------------------

     s/(s\np(sing,nom)):
        λvar32.[var32,ref8~person!] ▸
           [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]
   + s\np(sing,nom)/np(_,acc):
        λvar35.λvar36.[beat(pres,pres,perf,act,indic),var35,var36]
  → s/np(_,acc):
        λvar35.[beat(pres,pres,perf,act,indic),var35,ref8~person!] ▸
           [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey
ref7 may be bound to e2~brown~donkey (if e2 is consistent with
[male])
       or to e1~man~unkind
ref8 may be bound to e3~woman
       or to e1~man~unkind

>> him.
================================================
Next word: him
------------------------------------------------

     s/np(sing,acc):
        λvar35.[beat(pres,pres,perf,act,indic),var35,ref8~person!] ▸
           [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]
   + np(sing,acc)/nmod(sing,_):
        λvar37.[var37,ref10~male!] ▸ ref10~male!
  → s/nmod(sing,_129304):
        λvar37.[var37,ref10~male!] ▸
        [beat(pres,pres,perf,act,indic),ref10~male!,ref8~person!] ▸
           [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey~male
ref7 is bound to e2~brown~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~brown~donkey~male
```

```
================================================
End of Sentence

  ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),ref3~donkey~brown,ref1~man]
  ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]
  ** s:[beat(pres,pres,perf,act,indic),ref10~male!,ref8~person!] ▸
        [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~brown~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~brown~donkey~male
ref7 is bound to e2~brown~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~brown~donkey~male
------------------------------------------------------------------
∀x.(donkey(x) ⇒ nature(x,animal)) ∧
∀x.(male(x) ⇒ gender(x,masculine)) ∧
∀x.(man(x) ⇒ gender(x,masculine)) ∧ ∀x.(man(x) ⇒ type(x,human)) ∧
∀x.(man(x) ⇒ nature(x,animal)) ∧
∀x.(woman(x) ⇒ gender(x,feminine)) ∧
∀x.(woman(x) ⇒ type(x,human)) ∧ ∀x.(woman(x) ⇒ nature(x,animal)) ∧
∃v1.(woman(v1) ∧ ∃v2.(donkey(v2) ∧ male(v2) ∧ ∃v3.(man(v3) ∧
     unkind(v3) ∧ beat(v2,v3) ∧ feed(v2,v1) ∧
          beat(v2,v3) ∧ hate(v3,v2))))
------------------------------------------------------------------

>> ^D !! End of input Stream!!

 ** Terminating parse **

yes
```

4. Indexed Indefinite Reference

```
| ?- listen.
  ** George parser **

State 3:

  ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),ref3~donkey,ref1~man]
  ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]
  ** s:[beat(pres,pres,perf,act,indic),ref10~male!,ref8~person!] ▸
        [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~donkey~male
ref7 is bound to e2~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~donkey~male

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
>> The
================================================
Next word: The
------------------------------------------------

     np(sing,_126049)/nmod(sing,_126052)/n(sing):
        refl1#λvar38.[var38,refl1!] ▸ refl1!

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~donkey~male
ref7 is bound to e2~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~donkey~male
refl1 may be bound to e3~woman
       or to e2~donkey~male
       or to e1~man~unkind

------------------------------------------------

     np(plur,_126001)/nmod(plur,_126004)/n(plur):
        ref12#λvar39.∀ind3.[var39,ref12!×ind3] ▸ ref12!×ind3

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~donkey~male
ref7 is bound to e2~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~donkey~male
ref12 is bound to e3~woman
       and e2~donkey~male
       and e1~man~unkind

>> woman
================================================
Next word: woman
------------------------------------------------

     np(sing,_126557)/nmod(sing,_126560)/n(sing):
        refl1#λvar38.[var38,refl1!] ▸ refl1!
   + n(sing):woman
  → np(sing,_126557)/nmod(sing,_126560):
        λvar38.[var38,refl1~woman!] ▸ refl1~woman!

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~donkey~male
ref7 is bound to e2~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~donkey~male
refl1 is bound to e3~woman
```

```
>> gives
================================================
Next word: gives
------------------------------------------------

      np(sing,nom)/nmod(sing,_127089):
         λvar38.[var38,refl1~woman!] ▸ refl1~woman!
    + s\np(sing,nom)/pp(dir)/np(_127550,acc):
         λvar40.λvar41.λvar42.
            [give(pres,pres,perf,act,indic),var41,var40,var42]
   → s/pp(dir)/np(_127550,acc):
         λvar40.λvar41.
            [give(pres,pres,perf,act,indic),var41,var40,refl1~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~donkey~male
ref7 is bound to e2~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~donkey~male
ref11 is bound to e3~woman

------------------------------------------------

      np(sing,nom)/nmod(sing,_127089):
         λvar38.[var38,refl1~woman!] ▸ refl1~woman!
    + s\np(sing,nom)/np(_127497,acc)/np(_127500,acc):
         λvar43.λvar44.λvar45.
            [give(pres,pres,perf,act,indic),var43,var44,var45]
   → s/np(_127497,acc)/np(_127500,acc):
         λvar43.λvar44.
            [give(pres,pres,perf,act,indic),var43,var44,refl1~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~donkey~male
ref7 is bound to e2~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~donkey~male
ref11 is bound to e3~woman
```

```
>> the
================================================
Next word: the
------------------------------------------------
     s/pp(dir)/np(sing,acc):
        λvar40.λvar41.

[give(pres,pres,perf,act,indic),var41,var40,ref11~woman!]
   + np(sing,acc)/nmod(sing,_128482)/n(sing):
        ref13#λvar52.[var52,ref13!] ▸ ref13!
  → s/pp(dir)/nmod(sing,_128482)/n(sing):
        ref13#λvar52.λvar41.[var52,ref13!] ▸
              [give(pres,pres,perf,act,indic),var41,
                                        ref13!,ref11~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~donkey~male
ref7 is bound to e2~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~donkey~male
ref11 is bound to e3~woman
ref13 may be bound to e2~donkey~male
       or to e1~man~unkind

------------------------------------------------

     s/np(_127699,acc)/np(sing,acc):
        λvar43.λvar44.
           [give(pres,pres,perf,act,indic),var43,var44,ref11~woman!]
   + np(sing,acc)/nmod(sing,_128482)/n(sing):
        ref13#λvar52.[var52,ref13!] ▸ ref13!
  → s/np(_127699,acc)/nmod(sing,_128482)/n(sing):
        ref13#λvar52.λvar44.[var52,ref13!] ▸
           [give(pres,pres,perf,act,indic),ref13!,
                                        var44,ref11~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~donkey~male
ref7 is bound to e2~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~donkey~male
ref11 is bound to e3~woman
ref13 may be bound to e2~donkey~male
       or to e1~man~unkind

------------------------------------------------
```

```
      s/pp(dir)/np(plur,acc):
         λvar40.λvar41.
            [give(pres,pres,perf,act,indic),var41,var40,refl1~woman!]
    + np(plur,acc)/nmod(plur,_128434)/n(plur):
         ref14#λvar53.∀ind4.[var53,ref14!×ind4] ▸ ref14!×ind4
   → s/pp(dir)/nmod(plur,_128434)/n(plur):
         ref14#λvar53.λvar41.∀ind4.[var53,ref14!×ind4] ▸
            [give(pres,pres,perf,act,indic),var41,
                                        ref14!×ind4,refl1~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~donkey~male
ref7 is bound to e2~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~donkey~male
refl1 is bound to e3~woman
ref14 is bound to e2~donkey~male
       and e1~man~unkind

------------------------------------------------

      s/np(_127699,acc)/np(plur,acc):
         λvar43.λvar44.
            [give(pres,pres,perf,act,indic),var43,var44,refl1~woman!]
    + np(plur,acc)/nmod(plur,_128434)/n(plur):
         ref14#λvar53.∀ind4.[var53,ref14!×ind4] ▸ ref14!×ind4
   → s/np(_127699,acc)/nmod(plur,_128434)/n(plur):
         ref14#λvar53.λvar44.∀ind4.[var53,ref14!×ind4] ▸
            [give(pres,pres,perf,act,indic),ref14!×ind4,
                                          var44,refl1~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~donkey~male
ref4 is bound to e3~woman
ref5 is bound to e2~donkey~male
ref7 is bound to e2~donkey~male
ref8 is bound to e1~man~unkind
ref10 is bound to e2~donkey~male
refl1 is bound to e3~woman
ref14 is bound to e2~donkey~male
       and e1~man~unkind
```

```
>> donkey
================================================
Next word: donkey
------------------------------------------------

      s/pp(dir)/nmod(sing,_129823)/n(sing):
         ref13#λvar52.λvar41.[var52,ref13!] ▸

[give(pres,pres,perf,act,indic),var41,ref13!,ref11~woman!]
   + n(sing):donkey
  → s/pp(dir)/nmod(sing,_129823):
         λvar52.λvar41.[var52,ref13~donkey!] ▸
            [give(pres,pres,perf,act,indic),var41,
                                    ref13~donkey!,ref11~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey

------------------------------------------------

      s/np(_129430,acc)/nmod(sing,_129434)/n(sing):
         ref13#λvar52.λvar44.[var52,ref13!] ▸

[give(pres,pres,perf,act,indic),ref13!,var44,ref11~woman!]
   + n(sing):donkey
  → s/np(_129430,acc)/nmod(sing,_129434):
         λvar52.λvar44.[var52,ref13~donkey!] ▸
            [give(pres,pres,perf,act,indic),ref13~donkey!,
                                    var44,ref11~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
```

```
>> some
================================================
Next word: some
------------------------------------------------

     s/np(plur,acc)/nmod(sing,_130410):
        λvar52.λvar44.[var52,ref13~donkey!] ▸

[give(pres,pres,perf,act,indic),ref13~donkey!,var44,ref11~woman!]
   + np(plur,acc)/nmod(plur,_131225)/n(plur):
        ref15#λvar54.∀ind5.[var54,ref15×ind5] ▸ ref15×ind5
  → s/nmod(plur,_131225)/n(plur):
        ref15#λvar54.∀ind5.[var54,ref15×ind5] ▸
        [give(pres,pres,perf,act,indic),ref13~donkey!,
                                    ref15×ind5,ref11~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4

>> carrots
================================================
Next word: carrots
------------------------------------------------

     s/nmod(plur,_131373)/n(plur):
        ref15#λvar54.∀ind5.[var54,ref15×ind5] ▸
           [give(pres,pres,perf,act,indic),ref13~donkey!,
                                    ref15×ind5,ref11~woman!]
   + n(plur):carrot
  → s/nmod(plur,_131373):
        λvar54.∀ind5.[var54,ref15~carrot×ind5] ▸
           [give(pres,pres,perf,act,indic),ref13~donkey!,
                                ref15~carrot×ind5,ref11~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
```

```
================================================
End of Sentence

  ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),ref3~donkey,ref1~man]
  ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]
  ** s:[beat(pres,pres,perf,act,indic),ref10~male!,ref8~person!] ▸
          [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]
  ** s:∀ind5.[give(pres,pres,perf,act,indic),ref13~donkey!,
                                    ref15~carrot×ind5,ref11~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot

------------------------------------------------------------------
∀x.(carrot(x) ⇒ colour(x,orange)) ∧
∀x.(carrot(x) ⇒ nature(x,vegetable)) ∧
∀x.(donkey(x) ⇒ nature(x,animal)) ∧
∀x.(male(x) ⇒ gender(x,masculine)) ∧
∀x.(man(x) ⇒ gender(x,masculine)) ∧ ∀x.(man(x) ⇒ type(x,human)) ∧
∀x.(man(x) ⇒ nature(x,animal)) ∧
∀x.(woman(x) ⇒ gender(x,feminine)) ∧
∀x.(woman(x) ⇒ type(x,human)) ∧ ∀x.(woman(x) ⇒ nature(x,animal)) ∧
∃V1.((∀v1.v1∈V1 ⇒ carrot(v1)) ∧ ∃v2.(woman(v2) ∧
     ∃v3.(male(v3) ∧ donkey(v3) ∧ ∃v4.(man(v4) ∧
           unkind(v4) ∧ beat(v3,v4) ∧ feed(v3,v2) ∧ beat(v3,v4) ∧
                 hate(v4,v3) ∧ (∀v5.v5∈V1 ⇒ give(v3,v5,v2))))))
------------------------------------------------------------------

  !! End of input Stream !!

  ** Terminating parse **
```

5. Non-Introductory Indexed Reference

```
| ?- listen.
  ** George parser **

State 4:

  ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),ref3~donkey,ref1~man]
  ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]
  ** s:[beat(pres,pres,perf,act,indic),ref10~male!,ref8~person!] ▸
          [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]
  ** s:∀ind5.[give(pres,pres,perf,act,indic),ref13~donkey!,
                                    ref15~carrot×ind5,ref11~woman!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
>> The
================================================
Next word: The
------------------------------------------------

      np(sing,_)/nmod(sing,_)/n(sing):
         ref16#λvar55.[var55,ref16!] ▸ ref16!

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 may be bound to e3~woman
       or to e2~male~donkey
       or to e1~man~unkind

------------------------------------------------

      np(plur,_)/nmod(plur,_)/n(plur):
         ref17#λvar56.∀ind6.[var56,ref17!×ind6] ▸ref17!×ind6

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref17 is bound to e4~carrot
       and e3~woman
       and e2~male~donkey
       and e1~man~unkind
```

```
>> donkey
================================================
Next word: donkey
------------------------------------------------

      np(sing,_)/nmod(sing,_)/n(sing):
         ref16#λvar55.[var55,ref16!] ▸ ref16!
    + n(sing):donkey
   → np(sing,_)/nmod(sing,_):
         λvar55.[var55,ref16~donkey!] ▸ ref16~donkey!

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey

>> eats
================================================
Next word: eats
------------------------------------------------

      np(sing,nom)/nmod(sing,_):
         λvar55.[var55,ref16~donkey!] ▸ ref16~donkey!
    + s\np(sing,nom)/np(_,acc):
         λvar57.λvar58.[eat(pres,pres,perf,act,indic),var57,var58]
   → s/np(_,acc):
         λvar57.[eat(pres,pres,perf,act,indic),var57,ref16~donkey!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
```

```
>> the
================================================
Next word: the
------------------------------------------------

     s/np(sing,acc):
        λvar57.[eat(pres,pres,perf,act,indic),var57,ref16~donkey!]
   + np(sing,acc)/nmod(sing,_128677)/n(sing):
        ref18#λvar61.[var61,ref18!] ▸ ref18!
  → s/nmod(sing,_)/n(sing):
        ref18#λvar61.[var61,ref18!] ▸
           [eat(pres,pres,perf,act,indic),ref18!,ref16~donkey!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref18 may be bound to e3~woman
       or to e1~man~unkind

------------------------------------------------

     s/np(plur,acc):
        λvar57.[eat(pres,pres,perf,act,indic),var57,ref16~donkey!]
   + np(plur,acc)/nmod(plur,_128629)/n(plur):
        ref19#λvar62.∀ind7.[var62,ref19!×ind7] ▸
ref19!×ind7
  → s/nmod(plur,_128629)/n(plur):
        ref19#λvar62.∀ind7.[var62,ref19!×ind7] ▸
        [eat(pres,pres,perf,act,indic),ref19!×ind7,ref16~donkey!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot
       and e3~woman
       and e1~man~unkind
```

```
>> carrots.
================================================
Next word: carrots
------------------------------------------------

      s/nmod(plur,_)/n(plur):
         ref19#λvar62.∀ind7.[var62,ref19!×ind7] ▸

[eat(pres,pres,perf,act,indic),ref19!×ind7,ref16~donkey!]
   + n(plur):carrot
   → s/nmod(plur,_):
         λvar62.∀ind7.[var62,ref19~carrot!×ind7] ▸
            [eat(pres,pres,perf,act,indic),
                                    ref19~carrot!×ind7,ref16~donkey!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot
```

```
================================================
End of Sentence

  ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),ref3~donkey,ref1~man]
  ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]
  ** s:[beat(pres,pres,perf,act,indic),ref10~male!,ref8~person!] ▸
          [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]
  ** s:∀ind5.[give(pres,pres,perf,act,indic),ref13~donkey!,
                                    ref15~carrot×ind5,ref11~woman!]
  ** s:∀ind7.[eat(pres,pres,perf,act,indic),
                                    ref19~carrot!×ind7,ref16~donkey!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot

----------------------------------------------------------------
∀x.(carrot(x) ⇒ colour(x,orange)) ∧
∀x.(carrot(x) ⇒ nature(x,vegetable)) ∧
∀x.(donkey(x) ⇒ nature(x,animal)) ∧
∀x.(male(x) ⇒ gender(x,masculine)) ∧
∀x.(man(x) ⇒ gender(x,masculine)) ∧ ∀x.(man(x) ⇒ type(x,human)) ∧
∀x.(man(x) ⇒ nature(x,animal)) ∧
∀x.(woman(x) ⇒ gender(x,feminine)) ∧
∀x.(woman(x) ⇒ type(x,human)) ∧ ∀x.(woman(x) ⇒ nature(x,animal)) ∧
∃V1.(( ∀v1.v1∈V1 ⇒ carrot(v1)) ∧ ∃v2.(woman(v2) ∧
     ∃v3.(male(v3) ∧ donkey(v3) ∧
          ∃v4.(man(v4) ∧ unkind(v4) ∧ beat(v3,v4) ∧ feed(v3,v2) ∧
               beat(v3,v4) ∧ hate(v4,v3) ∧
                    (∀v5.v5∈V1 ⇒ give(v3,v5,v2)) ∧
                         (∀v6.v6∈V1 ⇒ eat(v6,v3))))))
----------------------------------------------------------------

  !! End of input Stream !!

  ** Terminating parse **
```

6. Indexed Reference to Unions of Sets

```
| ?- listen.
  ** George parser **

State 5:

  ** s:coref(ref2~unkind,ref1~man) ▸
[beat(pres,pres,perf,act,indic),ref3~donkey,ref1~man]
  ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]
  ** s:[beat(pres,pres,perf,act,indic),ref10~male!,ref8~person!] ▸
[hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]
  ** s:∀ind5.[give(pres,pres,perf,act,indic),ref13~donkey!,
                                    ref15~carrot×ind5,ref11~woman!]
  ** s:∀ind7.[eat(pres,pres,perf,act,indic),
                                    ref19~carrot!×ind7,ref16~donkey!]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
>> The
================================================
Next word: The
------------------------------------------------

      np(sing,)/nmod(sing,_)/n(sing):
         ref20#λvar63.[var63,ref20!] ▸ ref20!

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot
ref20 may be bound to e3~woman
       or to e2~male~donkey
       or to e1~man~unkind

------------------------------------------------

      np(plur,_)/nmod(plur,_)/n(plur):
         ref21#λvar64.∀ind8.[var64,ref21!×ind8] ▸ ref21!×ind8

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot
ref21 is bound to e4~carrot
       and e3~woman
       and e2~male~donkey
       and e1~man~unkind
```

```
>> people
================================================
Next word: people
------------------------------------------------

     np(plur,_126414)/nmod(plur,_126417)/n(plur):
        ref21#λvar64.∀ind8.[var64,ref21!×ind8] ▸ ref21!×ind8
   + n(plur):person
   → np(plur,_126414)/nmod(plur,_126417):
        λvar64.∀ind8.[var64,ref21~person!×ind8] ▸
                                          ref21~person!×ind8

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot
ref21 is bound to e3~woman
       and e1~man~unkind

>> ride
================================================
Next word: ride
------------------------------------------------

     np(plur,nom)/nmod(plur,_127688):
        λvar64.∀ind8.[var64,ref21~person!×ind8] ▸
                                       ref21~person!×ind8
   + s\np(plur,nom)/np(_128274,acc):
        λvar65.λvar66.[ride(pres,pres,perf,act,indic),var65,var66]
   → s/np(_128274,acc):
        λvar65.∀ind8.[ride(pres,pres,perf,act,indic),
                                       var65,ref21~person!×ind8]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot
ref21 is bound to e3~woman
       and e1~man~unkind
```

```
>> the
================================================
Next word: the
------------------------------------------------

     s/np(sing,acc):
        λvar65.∀ind8.[ride(pres,pres,perf,act,indic),
                                        var65,ref21~person!×ind8]
   + np(sing,acc)/nmod(sing,_129036)/n(sing):
        ref22#λvar69.[var69,ref22!] ▸ ref22!
  → s/nmod(sing,_129036)/n(sing):
        ref22#λvar69.∀ind8.[var69,ref22!] ▸
                            [ride(pres,pres,perf,act,indic),
                                        ref22!,ref21~person!×ind8]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot
ref21 is bound to e3~woman
       and e1~man~unkind
ref22 is bound to e2~male~donkey
```

```
-------------------------------------------------

     s/np(plur,acc):
        λvar65.∀ind8.[ride(pres,pres,perf,act,indic),
                                          var65,ref21~person!×ind8]
   + np(plur,acc)/nmod(plur,_)/n(plur):
        ref23#λvar70.∀ind9.[var70,ref23!×ind9] ▸ ref23!×ind9
  → s/nmod(plur,_)/n(plur):
        ref23#λvar70.∀ind8.∀ind9.[var70,ref23!×ind9] ▸
           [ride(pres,pres,perf,act,indic),
                                  ref23!×ind9,ref21~person!×ind8]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot
ref21 is bound to e3~woman
       and e1~man~unkind
ref23 is bound to e4~carrot
       and e2~male~donkey
```

```
>> donkey.
================================================
Next word: donkey
------------------------------------------------

     s/nmod(sing,_)/n(sing):
        ref22#λvar69.∀ind8.[var69,ref22!] ▸
           [ride(pres,pres,perf,act,indic),
                                   ref22!,ref21~person!×ind8]
   + n(sing):donkey
   → s/nmod(sing,_):
        λvar69.∀ind8.[var69,ref22~donkey!] ▸
           [ride(pres,pres,perf,act,indic),
                                   ref22~donkey!,ref21~person!×ind8]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot
ref21 is bound to e3~woman
       and e1~man~unkind
ref22 is bound to e2~male~donkey
```

```
================================================
End of Sentence

  ** s:coref(ref2~unkind,ref1~man) ▸
        [beat(pres,pres,perf,act,indic),ref3~donkey,ref1~man]
  ** s:[feed(pres,pres,perf,act,indic),ref5~donkey!,ref4~woman]
  ** s:[beat(pres,pres,perf,act,indic),ref10~male!,ref8~person!] ▸
          [hate(pres,pres,perf,act,indic),ref8~person!,ref7~male!]
  ** s:∀ind5.[give(pres,pres,perf,act,indic),ref13~donkey!,
                                    ref15~carrot×ind5,ref11~woman!]
  ** s:∀ind7.[eat(pres,pres,perf,act,indic),
                                    ref19~carrot!×ind7,ref16~donkey!]
  ** s:∀ind8.[ride(pres,pres,perf,act,indic),
                                    ref22~donkey!,ref21~person!×ind8]

ref2 and ref1 are bound to e1~man~unkind
ref3 is bound to e2~male~donkey
ref4 is bound to e3~woman
ref5 is bound to e2~male~donkey
ref7 is bound to e2~male~donkey
ref8 is bound to e1~man~unkind
ref10 is bound to e2~male~donkey
ref11 is bound to e3~woman
ref13 is bound to e2~male~donkey
ref15 is bound to e4~carrot
ref16 is bound to e2~male~donkey
ref19 is bound to e4~carrot
ref21 is bound to e3~woman
       and e1~man~unkind
ref22 is bound to e2~male~donkey

------------------------------------------------------------------------
∀x.(carrot(x) ⇒ colour(x,orange)) ∧
∀x.(carrot(x) ⇒ nature(x,vegetable)) ∧
∀x.(donkey(x) ⇒ nature(x,animal)) ∧
∀x.(male(x) ⇒ gender(x,masculine)) ∧
∀x.(man(x) ⇒ gender(x,masculine)) ∧ ∀x.(man(x) ⇒ type(x,human)) ∧
∀x.(man(x) ⇒ nature(x,animal)) ∧
∀x.(woman(x) ⇒ gender(x,feminine)) ∧
∀x.(woman(x) ⇒ type(x,human)) ∧ ∀x.(woman(x) ⇒ nature(x,animal)) ∧
∃V1.((∀v1.v1∈V1 ⇒ carrot(v1)) ∧ ∃v2.(woman(v2) ∧
     ∃v3.(male(v3) ∧ donkey(v3) ∧
          ∃v4.(man(v4) ∧ unkind(v4) ∧
               beat(v3,v4) ∧ feed(v3,v2) ∧
                    beat(v3,v4) ∧ hate(v4,v3) ∧
                         (∀v5.v5∈V1 ⇒ give(v3,v5,v2)) ∧
                              (∀v6.v6∈V1 ⇒ eat(v6,v3)) ∧
                                 ride(v3,v2) ∧ ride(v3,v4)))))
------------------------------------------------------------------------

  !! Save discourse state !!

  ** Terminating parse **
```

Appendix C
The George Rules

A new indefinite simple reference is always bound to a new entity token created for that purpose.

Rule 1: *Simple Indefinite Reference*

New sort information applied to a simple reference is passed to the entity token in its candidate set only when that set (and hence the binding) is singleton. Otherwise, any candidate entity tokens which are inconsistent with the new information are removed from the candidate set.

Rule 2: *Sort Refinement*

The initial candidate set of a new simple definite reference always initially contains all the entity tokens in the system which are both sort and number consistent with the reference.

Rule 3: *Simple Definite Reference*

1. **Simple references may not have as candidates entity tokens which are bound one-to-one to any indexed reference;**

2. **A binding involving an indexed reference must contain either**
   a) more than one entity token; or
   b) at least one entity token bound singleton to an indexed reference.

3. **Any two indexed references singleton bound to the same entity token must have equal upper bounds.**

Rule 4: *Number Consistency*

Let E be a candidate entity token for a reference, R, within a closed predicate containing other references R_i, not identical with R, then:

E is deleted from R's candidate set if:

1. **E is in a singleton binding with any of R_i; and**
2. **R's candidate set is not singleton.**

Rule 5: *Non-Co-Reference within Closed Predicates*

A post-modifier (or, in general, a clausal modifier) is considered to relate to information already presented in the discourse if and only if:

1a. There exists in the preceding discourse a sentence which expresses the information referred to by the context extension predicate – it has the same predicate and all corresponding references between the old and the new predicate are referentially consistent.

and 1b. The references in that preceding sentence are bound to entity tokens which are referentially consistent with the corresponding references in the context extension predicate.

or 2. Discourse world deduction allow us to generate an intermediate closed predicate from the context extension for which parts 1a and 1b above hold.

Rule 6: *Context Extension Matching*

A new indefinite indexed reference is always singleton bound to a new entity token created for that purpose.

Rule 7: *Indexed Indefinite Reference*

The initial candidate set of a new indexed definite reference always initially contains all the entity tokens in the system which are both sort consistent in GRL and number consistent in George (Rule 4) with the reference.

Rule 8: *Indexed Definite Reference*

Let E be an entity token bound to an indexed reference. Then:

Select new entity tokens E_1 ... E_n from Ent, apply to each the sort information applied to E, and insert them in the Entity Tokens Database. Add the pairs $\{\langle E, E_i\rangle \mid 1 \leq i \leq n\}$ to the partial order $\gg$.

The Entity Token Partition Tree of E is then defined as

$$\{E_i \mid 1 \leq i \leq n\} \cup \bigcup_{i \leq n} T_i$$

where $E \gg E_i$ and T_i is the Entity Token Partition Tree of E_i.

Rule 9: *Entity Token Partition*

Let $I_1...I_p$, $J_1...J_q$ be index symbols, I be a bounded index symbol with upper bound N, and K be an arbitrary, possibly empty, string of I_i and J_i conjoined by $\times$ and $\otimes$. Let R, $R_1...R_n$, $S_1...S_m$ be references (n, m, p, q, $N \in \mathbb{N}$), and P be a predicate symbol. Then:

We can rewrite an expression of the form

$$\forall I_1...\forall I_p.\forall I < N.\forall J_1...\forall J_q.[P,R_n...R_1,R \times I \times K,S_m...S_1]$$

by creating N of copies of the expression each with I replaced by a different member of $\mathbb{N}$ less than N, iff N is instantiated with a known value, thus:

$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R \times 0 \times K,S_m...S_1]$$
$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R \times 1 \times K,S_m...S_1]$$
$$\ldots$$
$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R \times (N\text{-}1) \times K,S_m...S_1]$$

After rewriting, each reference symbol with an integer applied to it is replaced by a new unique reference symbol; all occurrences of each ⟨reference symbol,integer⟩ pair are replaced by the same symbol. Weak index expansion causes a corresponding entity token partition (if one does not already exist), so each new reference has its own distinct entity token. Having done so, bindings containing the expanded reference must be replaced by n bindings between the new references and entity tokens.

Rule 10: *Weak Index Expansion*

Let I be an index symbol, R, $R_1...R_n$, $S_1...S_m$ be references (n, m $\in \mathbb{N}$), and P be a predicate symbol. Then:

Given an expression of the form

$$\forall I.[P,R_n,...,R_1,R \times I,S_m,...,S_1],$$

we can replace any number of R_i with $R_i \times I$ iff

1) **the index I only appears once within this closed predicate (ie it has never been propagated);**
2) **the Index Application Rule (Rule 18) holds for $\langle left,R_i,R \times I\rangle$.**

To maintain consistency, I must also be propagated to any identical occurrences of R_i in other expressions in the discourse memory. If appropriate, quantifiers should be added.

Rule 11: *Weak Index Propagation*

Let I_1 and I_2 be index symbols respectively, R be a simple reference, $R_1...R_n$, $S_1...S_m$ be references (n, m $\in \mathbb{N}$), and let P be a predicate symbol. Then:

To express index dependency, we can rewrite an expression of the form

$$\forall I_1.\forall I_2.[P,R_n...R_1,R \times I_2 \times I_1,S_m...S_1]$$

by replacing $R \times I_2 \times I_1$ with $R \times I_1$ iff (upb I_1 = upb I_2) to give

$$\forall I_1.[P,R_n...R_1,R \times I_1,S_m...S_1]$$

This gives rise to the new Bounding Constraint

$$\text{upb } I_1 = \text{upb } I_2$$

Rule 12: *Index Dependency*

Let I, $I_1...I_n$, $J_1...J_p$, $K_1...K_q$ be index symbols, R, $R_1...R_r$, $S_1...S_m$ be references (n, m, p, q, r $\in \mathbb{N}$), and P be a predicate symbol. Then:

We can rewrite an expression of the form

$$\forall J_1...\forall J_n.\forall I.\forall K_1...\forall K_q.[P,R_r,...,R_1,R \times I,S_m,...,S_1]$$

by dividing the domain of I into n parts, allowing new indices $I_1...I_n$ to range over them, and replacing the expression by n copies with I_i substituted for I, to give

$$\forall J_1...\forall J_n.\forall I_1.\forall K_1...\forall K_q.[P,R_r,...,R_1,R \times I_1,S_m,...,S_1]$$
$$\forall J_1...\forall J_n.\forall I_2.\forall K_1...\forall K_q.[P,R_r,...,R_1,R \times I_2,S_m,...,S_1]$$
$$...$$
$$\forall J_1...\forall J_n.\forall I_n.\forall K_1...\forall K_q.[P,R_r,...,R_1,R \times I_n,S_m,...,S_1]$$

This creates a new Bounding Constraint

$$\text{upb } I = \text{upb } I_1 + \text{upb } I_2 + ... + \text{upb } I_n$$

Whenever such an index partition is performed, a corresponding entity token partition must also be executed (if one does not already exist), and new bindings must be created to replace that of R × I.

Rule 13: *Weak Index Partition*

Let Q_1, Q_2 be (possibly empty) strings of quantifiers, I, $I_1...I_n$, J, $J_1...J_m$ be index symbols, R, $R_1...R_r$, S, $S_1...S_p$, $T_1...T_q$ be references (n, m, p, q, $r \in \mathbb{N}$), and P be a predicate symbol. Let k_i be in $\mathbb{N}$ for $1 \leq i \leq n$, such that $1 \leq k_i < m$. Then:

We can rewrite an expression of the form

$$Q_1.\forall I.\forall J.Q_2.[P,R_n,...,R_1,R \times I,S_p,...,S_1,S \times J,T_q...T_1]$$

iff the index application rule (Rule 19) holds of ⟨left,R × I,S × J⟩, as follows.

1. **Divide the domain of I into $I_1...I_n$ by weak index partition (Rule 19);**
2. **Divide the resulting expression containing I_1 by weak index partition of J into $J_1...J_m$;**
3. **Divide the domain of J in each remaining expression resulting from step 1, above, into the <u>same</u> index partition as in 2, replacing the expression by k_j copies with J_i substituted for J, to give (where $1 \leq i \leq n$ and $1 \leq k_j \leq m$)**

$$Q_1.\forall I_j.\forall J_1.Q_2.[P,R_n,...,R_1,R \times I_j,S_p,...,S_1,S \times J_1,T_q...T_1]$$
$$Q_1.\forall I_j.\forall J_2.Q_2.[P,R_n,...,R_1,R \times I_j,S_p,...,S_1,S \times J_2,T_q...T_1]$$
$$...$$
$$Q_1.\forall I_j.\forall J_{k_j}.Q_2.[P,R_n,...,R_1,R \times I_j,S_p,...,S_1,S \times J_{k_j},T_q...T_1]$$

New bindings must be created, to replace that of S × J, between $S \times J_2...S \times J_k$ and the 2nd...k_ith of the m new entity tokens, respectively.

The rule also applies in the reverse direction (ie with I and J interchanged).

Rule 14: *Compound Index Partition*

Let I be an index symbol, R, $R_1 ... R_n$, $S_1 ... Sm$ be references ($n, m \in \mathbb{N}$), and P be a predicate symbol. Then:

Given an expression of the form

$$\forall I.[P,R_n,...,R_1,R \otimes I,S_m,...,S_1],$$

all of R_i must be replaced with $R_i \otimes I$, and all of S_j with $S_j \otimes I$, iff

1) **the index I appears exactly once in the expression (ie it has not already been propagated); and**
2) **the Index Application Rule (Rule 19) holds for $\langle left,R_i,R \otimes I \rangle$ or $\langle right,S_j,R \otimes I \rangle$ as appropriate.**

To maintain consistency, I must also be propagated to any identical occurrences of R_i in other expressions in the discourse memory. If appropriate, quantifiers should be added.

Rule 15: *Strong Index Propagation*

Let $I_1...I_p$, $J_1...J_q$ be index symbols, I be a bounded index symbol with upper bound N, and K be an arbitrary, possibly empty, string of I_i and J_i conjoined by $\times$. Let R, $R_1...R_n$, $S_1...S_m$ be references (n, m, p, q, $N \in \mathbb{N}$), and P be a predicate symbol. Then:

Given an expression of the form

$$\forall I_1...\forall I_p.\forall I{<}N.\forall J_1...\forall J_q.[P,R_n...R_1,R \times K \otimes I,S_m...S_1]$$

we can creating N copies of the expression, each with I replaced by a different member of $\mathbb{N}$ less than N, thus:

$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R \times K \otimes 0,S_m...S_1]$$
$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R \times K \otimes 1,S_m...S_1]$$
$$...$$
$$\forall I_1...\forall I_p.\forall J_1...\forall J_q.[P,R_n...R_1,R \times K \otimes (N\text{-}1),S_m...S_1]$$

Then, if R is definite, we must partition its bound entity token so that there is a new, unique token bound to <u>each</u> copy of R; if R is indefinite, we must bind each new reference to the <u>single</u> existing entity token.

After this rewriting, each reference symbol with an integer applied to it must be replaced by a new reference symbol; all occurrences of each ⟨reference symbol,integer⟩ pair are replaced by the same symbol.

Rule 16: *Strong Index Expansion*

Let I, $I_1...I_k$, $J_1...J_p$, $K_1...K_q$ be index symbols, R, $R_1...R_n$, $S_1...S_m$ be references ($n, m, p, q \in \mathbb{N}$), and P be a predicate symbol. Then:

We can rewrite an expression of the form

$$\forall J_1...\forall J_n.\forall I_1.\forall K_1...\forall K_q.[P,R_n,...,R_1,R\otimes I,S_m,...,S_1]$$

by dividing the domain of I into k parts, allowing new indices I_1 ... I_k to range over them, and replacing the expression by n copies with I_1 ... I_k respectively substituted for I, to give

$$\forall J_1...\forall J_n.\forall I_1.\forall K_1...\forall K_q.[P,R_n,...,R_1,R\otimes I_1,S_m,...,S_1]$$
$$\forall J_1...\forall J_n.\forall I_2.\forall K_1...\forall K_q.[P,R_n,...,R_1,R\otimes I_2,S_m,...,S_1]$$
$$...$$
$$\forall J_1...\forall J_n.\forall I_k.\forall K_1...\forall K_q.[P,R_n,...,R_1,R\otimes I_k,S_m,...,S_1]$$

This creates a new Bounding Constraint

$$\text{upb } I = \text{upb } I_1 + \text{upb } I_2 + ... + \text{upb } I_n$$

Whenever a strong index partition is performed upon a reference R, a corresponding entity token partition must also be executed on the token bound to it.

Rule 17: *Strong Index Partition*

Let R_i be references and I_j be indices. Let P be a predicate symbol, and D be a direction, such that $D \in \{ \text{left, right} \}$. Then:

An index may be propagated in a direction D from R_1 to R_2 (denoted by $\langle D,R_1,R_2 \rangle$) unless precluded by one of the following:

1. **No index may propagate on to a strongly indexed reference.**
2. **No weak index may propagate on to a simple reference.**
3. **No strong index may propagate on to a definite reference.**

Rule 18: *Index Application*

Let E2 ▸ E1 be a context extended expression. Let R1 be a reference in E1, and R2 a reference in E2, both before index propagation. Then:

If the set of entities sort consistent with R1 is a superset of the set of entities sort consistent with R2, R1 may optionally be indexed with any strong index associated with or propagating on to R2. The candidate set of R1 is then reduced to be the same as that of R2.

Rule 19: *Index Propagation through Context Extension*

Appendix D
BNF Specification of GRL Syntax

1. Introduction

There follows a specification in Backus-Naur Form of the syntax of the George Representation Language. All terms are defined in Chapter 4. Non-terminal symbols are underlined; terminal symbols are **emboldened**. Vertical bar (|) denotes disjunction in the syntactic specification; juxtaposition denotes concatenation; postfixed * means zero or more repetitions; postfixed + means one or more repetitions (as in Kleene * and +). Curly braces indicate grouping in the description language (so parentheses and square brackets are always terminal symbols). Note that "Sentence" and "Quantified-Expression" are interchangeable. Where non-terminal symbols are modified with arguments (*eg* Index-Symbol(n)), the argument is taken to be constant throughout each production.

This specification, like the definition in Chapter 4, is in two parts – first, there is the basic expression structure, and then there is the quantifier expansion operation which constructs more convenient and conventional expressions out of the basic ones.

2. Basic Expression Specification

Expression	::=	Co-Reference | Closed-Predicate
Closed-Predicate	::=	[Predicate { , Reference }$^+$]
Predicate	::=	Predicate-Symbol(Modifiers)
Predicate-Symbol	::=	a member of Pred
Modifiers	::=	a member of $\Pi_{m \in Mod} Dom(m)$
Co-Reference	::=	**coref**(Reference , Reference)

Reference	::=	Simple-Reference |
		Indexed-Reference |
		Context-Extended-Reference
Simple-Reference	::=	Definite-Reference |
		Indefinite-Reference
Definite-Reference	::=	Indefinite-Reference !
Indefinite-Reference	::=	Reference-Symbol |
		Indefinite-Reference ˜ Sort-Symbol
Indexed-Reference	::=	Indexed-Reference |
		Index-Propagated-Reference
Indexed-Reference	::=	Index-Declaration(n) . Reference ⊗ Index(n) |
		Index-Declaration(n) . Reference × Index(n)
Index-Propagated-Reference	::=	Reference ⊗ Index(n) |
		Reference × Index(n)
Reference-Symbol	::=	**ref**n
Index-Declaration(n)	::=	∀ Index-Symbol(n) |
		∀ Index-Symbol(n) < Number |
		∀ Index-Symbol(n) < ⊥
Index-Symbol(n)	::=	**ind**n
Context-Extended-Reference	::=	Context-Extension ▸ Reference
Context-Extension	::=	Expression
n	::=	a member of ℕ
Sort-Symbol	::=	a member of Sort

3. Quantifier Expansion

Sentence	=	Quantified-Expression
Quantified-Expression	::=	Expansion(Expression)

where Expansion is an operation which moves Index-Declarations and Context-Extensions to the front of the argement Expression, yielding a syntactic object of the form:

{ Index-Declaration . }* { Context-Extension ▸ }+ Expression